Gh. Sabău
V. Avram
A. Cojocaru
*

A. Sotir
V. Orbeanu
C. Baronide
R. Nedelcu
**

N. Fildan
A. Alexandrescu
A. Dobrițoiu
D. V. Norea
**

Coordonatori : Gh. Sabău (coord. gen.) și A. Sotir

Practica bazelor de date

Totul despre... SOCRATE și SOCRATE-MINI pe Felix C, CORAL, Independent

1

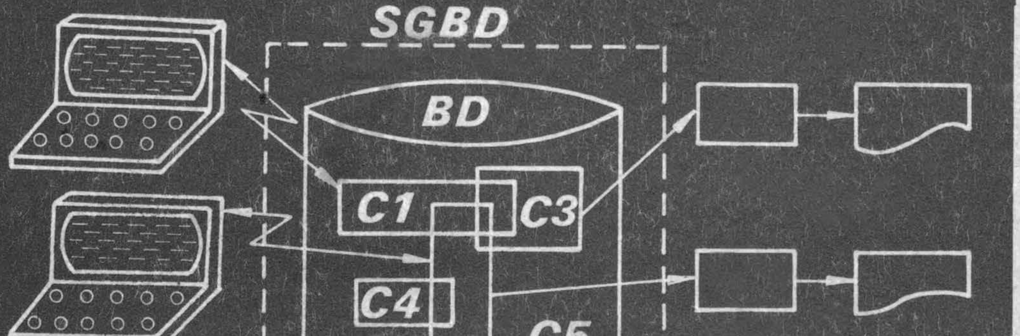
- Funcționare
- Dezvoltări
- Exemple

- Teste
- Aplicații complexe

AUTOMATICA ELECTRONICA

SERIA PRACTICĂ

INFORMATICĂ MANAGEMENT



AUTOMATICA

INFORMATICA

ELECTRONICA

SERIA PRACTICĂ

MANAGEMENT

Seria PRACTICĂ

BIBLIOTECA DE AUTOMATICĂ INFORMATICĂ-ELECTRONICĂ-MANAGEMENT

S. Radu, D. Filoti : Centrale telefonice automate
M. Bodea ş.a. : Tranzistoare cu efect de cîmp
D. N. Şapiro : Proiectarea radioreceptoarelor
V. Antonescu, M. Popovici : Ghid pentru controlul statistic al calităţii
V. Baltac ş.a. : Calculatorul FELIX C-256. Structură şi programare
G. Sonea, Sileţchi M. : Creşterea planificată a productivităţii muncii
R. L. Moriss : Proiectarea cu circuite integrate TTL
A. Brillianiov : Calculul şi construcţia televizoarelor portabile
Kaoru Ishikawa : Controlul de calitate pentru maiştri şi şefi de echipe
Magnus Radke : 222 măsuri pentru reducerea costurilor
I. Stăncioiu : Eficienţa economică a asimilării de utilaje noi
G. Lajtha : Proiectarea reţelelor de telecomunicaţii
Vătăşescu, A. ş.a. : Dispozitive semiconductoare. Manual practic
Ch. Jones : Design. Metode şi aplicaţii
E. S. Buffa : Conducerea modernă a producţiei, vol. I şi II
D. W. Davies, ş.a. : Reţele de interconectarea calculatoarelor
Gh. Baştiurea ş.a. : Comanda numerică a maşinilor-unelte
L. W. Crum : Analiza valorii
P. Foiaş : Automatica şi informatica în procesele editorial-poligrafice
P. Vezeanu, Şt. Pătraşcu : Măsurarea temperaturii în tehnică
T. Penescu, V. Petrescu : Măsurarea presiunii în tehnică
P. Popescu, P. Mihordea : Măsurarea debitului de tehnică
P. Vezeanu : Măsurarea nivelului în tehnică
A. Nadolo : Măsurarea volumului şi cantităţii lichidelor în industri
C. Hidoş, P. Isac (coordonatori) : Studiul muncii, I—VIII
C. Hidoş : Analiza şi proiectarea circuitelor informaţionale
Gh. I. Pisău : Elaborarea şi implementarea sitemelor informatice
P. Constantinescu, ş.a. : Sisteme informatice, modele ale conducerii
V. Penescu, ş.a. : Fişiere, baze şi bănci de date
I. Ceauşu ş.a. : S.D.V. Organizarea concepţiei, fabricaţiei, gestiunii
S. Brebenel : Practica transferului internaţional de tehnologie
P. Constantinescu ş.a. : Analiză, decizie, control
A. Vătăşescu ş.a. : Circuite integrate liniare. Manual de utilizare, vol. 1, 2, 3
S. Maican : Sisteme numerice cu circuite integrate
I. Ristea ş.a. : Manualul muncitorului electronist
M. Florescu ş.a. : Cibernetică, informatică, automatică în industria chimică
E. Statnic, M. Gănescu : Televizoare cu circutie integrate
T. Geber ş.a. : Echipamente periferice
S. Călin ş.a. : Optimizări în automatizări industriale
M. Simonescu : Proiectarea unitară a circuitelor electronice
C. Cruceru : Tehnica măsurărilor în telecomunicaţii
P. Niţulescu : Electroalimentarea instalaţiilor de telecomunicaţii
R. Râpeanu ş.a. : Circuite integrate analogice. Catalog
T. Rădulescu ş.a. : Centrale telefonice automate
N. Drăgulănescu : Agenda radioelectronistului
V. Baltac, A. Davidoviciu, C. Maşec ş.a. : Sisteme interactive şi limbaje conversaţionale
Th. Borangiu, R. Dobrescu : Structuri moderne de conducere automată a maşinilor unelte
A. Petrescu ş.a. : Microcalculatoarele FELIX M18, M18B, M118, vol. I şi II
D. Stanomir : Sisteme electroacustice
N. Iosif ş.a. : Tiristoare şi module de putere. Catalog
S. Călin, I. Dumitrache ş.a. : Reglarea numerică a proceselor tehnologice
G. Ionescu ş.a. : Traductoare pentru automatizări industriale
D. Boboc ş.a. : Cartea operatorului de la tablourile de automatizare
A. Millea : Cartea metrologului
M. Voicu : Tehnici de analiză a stabilităţii sistemelor automate
A. Stănescu ş.a. : Sisteme de automatizare pneumatice
H. M. Moţit, A. Ciocîrlea-Vasilescu : Debitmetrle industrială
N. Tertişco, P. Stoica, Th. Popescu : Indentificarea asistată de calculator a sistemelor
T. Baron ş.a. : Calitate şi fiabilitate. Manual practic, vol. 1+vol. 2

Lector univ. dr. ec. **Gheorghe Sabău**
Ec. **Vasile Avram**
Ec. **Aurelian Cojocaru**
Ing. **Alexandru Sotir**
Ing. **Valeriu Orbeanu**
Ing. **Constantin Baronide**
Ing. **Răzvan Nedelcu**
Ing. **Nicolae Fildan**
Ing. **Adrian Alexandrescu**
Mat. **Andrian Dobrițoiu**
Ing. **Dan Viorel Norea**

*　　**　　**

Practica bazelor de date

Totul despre... SOCRATE și SOCRATE-MINI pe Felix C, CORAL, INDEPENDENT

Coordonatori : **Gh. Sabău** (coord. gen.) și **Al. Sotir**

Vol. 1

- **Funcționare SOCRATE**
- **Dezvoltări SGBD-uri**
- **Exemple, teste**
- **Aplicații complexe**

EDITURA TEHNICĂ
București, 1989

Recenzii : Dr. mat. **MARGARETA DRĂGHICI**
Dr. ing. **STELIAN GUȚU**

Cuvînt înainte : Dr. mat. **MARGARETA DRĂGHICI**

Redactor : ing. **PAUL ZAMFIRESCU**

Tehnoredactor : **MARIA TRĂZNEA**
Desene : Arh. **MIOARA HORTOPAN**
ADRIAN GUGA

B.T. 27.02.1989. C.T. 31
C.Z. : 621.377

ISBN 973-31-0020-X
ISBN 973-31-0021-6

I. P. Oltenia – Craiova
Str. Mihai Viteazul nr. 4
Comanda 150/1989

CUVÎNT ÎNAINTE

Editura Tehnică pune la dispoziția cititorului român, așa cum ne-a obișnuit, o lucrare de deosebit interes.

Lucrarea se remarcă în primul rînd prin problematica abordată.

A ști să proiectezi și să administrezi baza de date a unui sistem nu mai poate fi, la ora actuală, o taină rezervată cîtorva aleși. Ritmul în care tehnica electronică de calcul pătrunde în toate sferele de activitate — întreprindere economică, proiectare, cercetare, medicină, învățămînt, administrație, servicii — multiplică cerința socială de a avea specialiști pregătiți să construiască baze de date. Avem nevoie de mulți specialiști.

Pe de altă parte, „specialistul" de astăzi are alt bagaj de cunoștințe față de specialistul de acum douăzeci de ani sau de acum zece ani (după cum avem certitudinea că „specialistul" de peste cinci ani va avea altă pregătire decît cel actual). Știința calculatoarelor, și ea într-o continuă evoluție, transferă din ce în ce mai multe activități de sub semnul artei sub cel al tehnicii. Capacitatea unui om de a proiecta bine sisteme — pentru că s-a depășit stadiul experimentelor cu largi toleranțe la risc — este determinată de măsura în care stăpînește metodologiile, tehnologiile, tehnicile, instrumentele existente. Prin problematica abordată volumul de față atrage atenția asupra aspectelor practice ale bazelor de date, ceea ce reprezintă un merit incontestabil al lucrării.

Meritorie este și ideea de a pune la îndemîna cititorului un volum axat pe însușirea sistemului SOCRATE — cel mai răspîndit sistem de gestiune a bazelor de date la noi în țară.

Sînt prezentate conceptele SOCRATE, care permit înțelegerea modelului în sine, specificul implementării SOCRATE pe Felix C, cît și toate manualele SOCRATE-MINI pe minisistemele românești.

...Din nou un merit al autorilor care îndeamnă cititorul către stăpînirea esenței lucrurilor și îi sugerează că a trece de la un tip de calculator la altul nu dublează efortul.

Mai menționăm cele două consistente aplicații de proiectare a bazelor de date : cea pentru activitatea de personal pe SOCRATE-Felix C și cea pentru activitatea de exploatare a flotei maritime, pe SOCRATE-MINI.

În sfîrșit, lucrarea este interesantă prin faptul că reprezintă rodul colaborării dintre cadre didactice, realizatori de sisteme de gestiune a bazelor de date și proiectanți de sisteme informatice cu baze de date. Experiențele prețioase dobîndite în cele trei tipuri de activități se conjugă cu experiența editorilor spre profitul cititorului.

Dr. mat. MARGARETA DRĂGHICI

NOTELE EDITURII

Despre autori :

Volumul 1

SABĂU GHEORGHE — lect. univ. dr. la Catedra de Cibernetică Economică din A.S.E., Bucureşti, la disciplina „Sisteme informatice şi bănci de date".

AVRAM VASILE — analist principal III în cadrul Centrului de Calcul al C.S.P., cadru didactic asociat la Facultatea de Planificare şi Cibernetică Economică din A.S.E.

COJOCARU AURELIAN — analist principal III la ICSIT-TCI, cadru didactic asociat la Facultatea de Planificare şi Cibernetică Economică, Bucureşti.

Volumul 2

Drd. ALEXANDRU SOTIR, absolvent al Facultăţii de electrotehnică a Institutului Politehnic Iaşi, coordonează, ca director tehnic, activitatea de exploatare a tehnicii de calcul la C.T.C.E. Constanţa.

Ing. NICOLAE FILDAN, absolvent al Facultăţii de electrotehnică-calculatoare a Institutului Politehnic Traian Vuia Timişoara, şef secţie Exploatare în cadrul C.T.C.E. Constanţa.

Ing. VALERIU ORBEANU, absolvent al Facultăţii Electronică şi Tc a Institutului Politehnic Bucureşti, şef secţie Exploatare în cadrul C.T.C.E. Constanţa.

Ing. ADRIAN ALEXANDRESCU, absolvent al Facultăţii de automatică-calculatoare a Institutului Politehnic Bucureşti, inginer de sistem pr. III la C.T.C.E. Constanţa, în cadrul colectivului de proiectare S.G.B.D. SOCRATE-MINI.

Ing. CONSTANTIN BARONIDE, absolvent al Facultăţii de automatică-calculatoare a Institutului Politehnic Bucureşti, inginer de sistem pr. III la C.T.C.E. Constanţa, în cadrul colectivului de proiectare S.G.B.D. SOCRATE-MINI.

Mat. ANDRIAN DOBRIŢOIU, absolvent al Facultăţii de matematică-mecanică a Universităţii Al. I. Cuza din Iaşi, programator pr. II la C.T.C.E. Constanţa în cadrul colectivului de proiectare S.G.B.D. SOCRATE-MINI.

Ing. RĂZVAN NEDELCU, absolvent al Facultăţii de automatică-calculatoare a Institutului Politehnic Bucureşti, inginer de sistem pr. III la C.T.C.E. Constanţa, în cadrul colectivului de proiectare S.G.B.D. SOCRATE-MINI.

Ing. DAN VIOREL NOREA, absolvent al Facultăţii de automatică-calculatoare a Institutului Politehnic Bucureşti, inginer de sistem pr. II la C.T.C.E. Constanţa, în cadrul colectivului de proiectare S.G.B.D. SOCRATE-MINI.

Despre carte

De mult timp Editura Tehnică era datoare cititorilor cu un ciclu de cărţi privind bazele de date. A sosit acum momentul să inaugurăm acest ciclu şi sîntem bucuroşi că prilejul ne-a fost oferit de autori români, cadre didactice, proiectanţi de software de bază şi de software de aplicaţii, informaticieni (economişti, ingineri, matematicieni) profesionişti.

În această lucrare s-au întîlnit preocupările a două grupuri de specialişti : primul compus din cadre didactice de la Catedra de Cibernetică Economică din Academia de Studii Economice, cu realizări în analiza şi proiectarea de aplicaţii informatice, împreună cu analişti de sistem de la Institutul de Tehnică de Calcul şi Informatică şi de la Comitetul de Stat al Planificării, care au oferit o lucrare privind SGBD-ul SOCRATE—pe Felix C ; cel de al doilea alcătuit din 8 spe-

cialişti de la Centrul de Calcul Teritorial Constanţa, elaboratori ai produsului software SOCRATE-MINI pentru minisistemele CORAL/INDEPENDENT, cărora redacţia le-a solicitat scrierea manualelor de lansare-utilizare-operare şi a unei aplicaţii complexe pe SOCRATE-MINI. În mod corespunzător, redacţia a solicitat şi primului colectiv să elaboreze o aplicaţie amplă pe SOCRATE Felix-C.

Pentru a atinge obiectivele propuse şi anume realizarea unei cărţi practice asupra conceptelor bazelor de date, asupra proiectării structurii bazelor de date şi a aplicaţiilor cu SGBD-uri SOCRATE, s-a încredinţat lect. univ. dr. ec. Gheorghe Sabău coordonarea generală a cărţii, iar drd. ing. Alexandru Sotir — director tehnic al CTCE-Constanţa participarea la această coordonare, din partea colectivului constănţean.

Redacţia a colaborat permanent pentru structurarea şi completarea ediţiei, în aşa fel încît cartea rezultată să fie unitară, să se adreseze unui cerc larg de cititori, atît specialişti în proiectarea aplicaţiilor cu baze de date cît şi cadre didactice, studenţi, elevi informaticieni, ciberneticieni, matematicieni, ingineri, economişti care lucrează în aplicaţii cu baze de date.

Datorită dotării noastre timpurii cu calculatoare Felix, compatibile cu familia franceză IRIS 50 şi a adoptării fireşti a sistemului corespunzător de gestiune a bazelor de date, de tip reţea, SOCRATE, autorii — care împărtăşesc părerea că SGBD-urile relaţionale sînt cele care vor domina în viitor aplicaţiile — au fost nevoiţi să prezinte un sistem nerelaţional, în mod sigur mult mai complicat, cu operatori şi structuri mai complexe şi relativ mai puţin funcţional. Acesta este, însă, SGBD-ul cu care se lucrează în întreaga ţară, un aspect esenţial ce nu putea fi neglijat. Şi, mai trebuie arătat că sistemele nerelaţionale (ierarhice, reţea) domină încă numeric pe cele relaţionale în implementările din întreaga lume, prezentînd, în unele situaţii, anumite avantaje. Această situaţie va mai persista, chiar dacă cercetările şi dezvoltările curente sînt bazate pe concepte relaţionale, iar ritmul instalării de produse relaţionale este extrem de ridicat ; de fapt toate noile sisteme sînt relaţionale sau „cvasirelaţionale". Pe de altă parte modelul relaţional este perfect adaptat cerinţelor didactice ; datele sînt percepute de utilizator ca tabele (şi numai ca tabele) iar operatorii aflaţi la dispoziţie (de exemplu pentru regăsirea datelor) generează noi tabele din cele vechi ; utilizatorul unui model nerelaţional (ierarhic, reţea) lucrează cu alte structuri de date (de exemplu arborescente), care trebuie manipulate de operatori mai complicaţi.

Cititorul nu trebuie, însă, să fie speriat : odată ce va învăţa SGBD-ul SOCRATE, el va reuşi să abordeze cu uşurinţă şi un model relaţional. Iar pentru deprinderile practice, SOCRATE şi SOCRATE MINI au avantajul a fi implementate pe toate calculatoarele Felix C şi, respectiv, pe minicalculatoarele CORAL/INDEPENDENT (ca şi pe toate minicalculatoarele compatibile cu PDP 11), pe cînd produsele relaţionale cum ar fi d'BASE 2, d'BASE 3, ORACLE ş.a. sînt încă în faza de pătrundere în unităţile noastre de calcul (pornind mai ales de la calculatoarele personal-profesionale, microcalculatoare şi minicalculatoare).

Prin organizarea cărţii, care admite o redundanţă controlată, cititorul are posibilitatea de a începe instruirea, fie de la secţiunea I — dacă doreşte o instruire consistentă, mai completă dar mai dificilă, care atinge pînă şi probleme complexe — cum sînt cele ale optimizărilor, fie de la secţiunea a V-a, dacă este mai obişnuit cu documentaţiile produselor-program specifice firmelor.

Exemple rezolvate sînt „presărate" în ambele volume, dar nu numai atît : cele două aplicaţii complexe simulează perfect proiectarea schemelor conceptuale ale bazelor de date ca şi manipularea datelor, pînă la rezolvarea unor probleme practice. Mai menţionăm şi testele incluse la începutul şi sfîrşitul volumului I, cu care cititorul îşi poate controla singur stadiul cunoştinţelor.

Adăugăm că în timp ce lucrarea se afla în tipografie a apărut versiunea 4.0 a produsului SOCRATE-MINI. Noutăţile acestei versiuni sînt incluse în volumul II, la paginile 256—270.

Credem că, pe drept, cărţile se subintitulează „Totul despre...", dar aceasta rămîne la aprecierea cititorilor, ale căror opinii le aşteptăm cu încredere.

CUPRINS GENERAL

Volumul 1

Cuvînt înainte 5
Notele editurii 7
Teste (1—8) 15

Secţiunea I. Baze de date şi SGBD-ul SOCRATE 17

1. Baze de date—definire şi caracteristici 17
2. SGBD SOCRATE. Prezentare sintetică 32

Secţiunea II. SGBD SOCRATE — Tratare generală 66

3. Limbajul de descriere a datelor 66
4. Limbajul de manipulare a datelor 115
5. Alte componente ale SGBD-SOCRATE 193
6. Utilizarea funcţiunilor SGBD-SOCRATE 238
7. Integritatea bazelor de date . . 274

Secţiunea III. Optimizări şi dezvoltări de SGBD-uri . . . 286

8. Optimizarea structurilor bazelor de date şi a programelor de exploatare 286
9. Dezvoltări SOCRATE şi alte SGBD-uri 348

Secţiunea IV. Aplicaţie complexă cu SGBD SOCRATE pentru activitatea de personal

10. Proiectarea bazei de date pentru activitatea de personal 375
11. Implementarea bazei de date PERSONAL (listing complet generarea bazei, întreţinerea bazei de date, generarea automată a programelor... testarea coerenţei fizice şi logice, simularea rulării) 393

ANEXE 455
Teste (9—25) 482
Bibliografie 495

Volumul 2

Secţiunea V. Manual de lansare SOCRATE-MINI

1. Introducere 11
2. Instalarea produsului 16
3. Noutăţile vesiunii 17
4. Portabilitatea versiunii 21
5. SOCRATE-MINI în „cifre" . . . 22
6. Anomalii şi restricţii ridicate în raport cu V.2.1 23
7. Restricţiile versiunii 24

Secţiunea VI. Manual de utilizare SOCRATE-MINI

1. Introducere 25
2. Limbajul de descriere a structurii 27
3. Limbajul de manipulare 50
4. Programul bibliotecar 83
5. Interfeţe de comunicare cu alte limbaje, exemple de programe SOCRATE, COBOL, PASCAL, FORTRAN, BASIC 92
6. Interfaţa pentru lucrul într-o reţea de baze de date, exemple de programe SOCRATE, FORTRAN, COBOL 105
7. Interfaţa SOCRATE-MINI-SOCRATE-Felix 123

Secţiunea VII. Manual de operare SOCRATE-MINI

1. Introducere 148
2. DFS — Procesorul de definire a structurii 151
3. SOC — Procesorul limbajului de manipulare 158
4. MGS — Programul bibliotecar . 163
5. Programe utilitare 167
6. Interfeţe de comunicare cu alte limbaje 194

7. Interfaţa pentru lucrul într-o reţea de baze de date 195
8. Interfaţa SOCRATE-MINI-SOCRATE Felix 197

Secţiunea VIII. Aplicaţie complexă cu SGDB SOCRATE-MINI pentru activitatea de exploatare a flotei maritime

1. Analiza activităţii 200
2. Proiectarea schemei conceptuale a bazei de date 204
3. Elaborarea programelor şi a procedurilor 209
4. Rezolvarea pe baza de date implementată, a activităţii de programare a transportului de mărfuri (Studiu de caz) 209
5. Listinguri complete pentru schema conceptuală, manipularea datelor rezolvarea studiului de caz . . . 246
ANEXĂ. Noutăţile versiunii V4.0 a produsului SOCRATE-MINI . . 257

CUPRINS DETALIAT

Volumul 1

Cuvînt înainte 5
Prezentări 7
Teste (1–8) 15

Secțiunea I. Baze de date și SGBD-ul SOCRATE

1. BAZE DE DATE-DEFINIRE ȘI CARACTERISTICI 17
   - Ce este o bază de date? 17
   - De ce baze de date? 18
   - Independența datelor 19
   - Arhitectura unei baze de date . 20
   - Nivele de structurare a datelor . 20
   - Sistemul de gestiune a bazei de date (SGBD) 22
   - Utilizatorii bazei de date 23
   - Administratorul bazei de date . . 25
   - Modele de date și compararea lor 28
   - Concluzii 32
2. SGBD SOCRATE — PREZENTARE SINTETICĂ 32
   - Definire și componente 32
   - Istoric și versiuni 34
   - Limbajul de descriere a datelor . 36
   - Limbajul de manipulare a datelor 41
   - Alte componente 50
     - Macrogeneratorul 50
     - Editorul de texte 53
     - Interfața cu limbaje evoluate . 53
   - Structuri de date acceptate de SGBD SOCRATE 54
   - Ciclul de viață al unei baze de date 57
   - Implementări ale dicționarului datelor 62
   - Reguli de prezentare a limbajelor 63
   - Concluzii 65

Secțiunea II. SGBD SOCRATE — tratare generală

3. LIMBAJUL DE DESCRIERE* A DATELOR. LDD-SOCRATE . . 66
   - Introducere 66
   - Caracteristici 69
     - Caracteristica de tip ⟨cuvînt⟩ (MOT) 69
     - Caracteristica de tip ⟨listă de valori⟩ 71
     - Caracteristica de tip ⟨valoare numerică⟩ 72
     - Caracteristica de tip ⟨text⟩ (TEXTE) 74
     - Caracteristica de tip ⟨bloc⟩ . 75
     - Caracteristica de tip ⟨entitate⟩ (ENTITE) 77
     - Caracteristica de tip ⟨inel⟩ (ANNEAU) 79
     - Caracteristica de tip ⟨referire⟩ (REFERE) 81
     - Caracteristica de tip ⟨invers⟩ (INVERSE) 84
     - Caracteristica de tip ⟨formal⟩ (FORMAL) 87
   - Tipuri de declarații de acces . . 90
   - Compilarea textelor de definire 94
     - Construirea dicționarului . . 94
     - Conținutul dicționarului . . . 95
   - Modificări ale LDD-SOCRATE în versiunea VI.6.R-I.T.C.I. . . . 95
     - Caracteristica de tip zecimal (DECIMAL) 96
     - Modificarea caracteristicilor din formal 96
     - Caracteristici de tip ⟨formal pentru sortare⟩ (FORMAL SORT) 98
     - Schimbarea modului de acces la dicționar 100
     - Definirea dinamică a cheilor de acces 100
   - Exemplu complex de definire a unei baze de date 102
   - Concluzii 114
4. LIMBAJUL DE MANIPULARE A DATELOR. LMD-SOCRATE . . 115
   - Tipuri de cereri 115
   - Citație 118
   - Calificare 118
     - Valabilitatea calificărilor . . . 119
     - Calificator de tip ⟨bloc⟩ . . . 122
     - Calificator de tip ⟨entitate⟩/⟨inel⟩/⟨invers⟩ 122

* Se numește și „limbaj de definire".

Calificator de tip ⟨referire⟩ 123
Calificator de tip variabilă Xi 124
Calificator X0 128
Condiţii 129
Sintaxa de definire a cererilor condiţionale 129
Evaluarea cererilor condiţionale 131
Noţiunea de filtru 134
Noţiunea de acces 136
Acces secvenţial 136
Acces secvenţial ordonat . . 137
Acces prin inel 137
Acces direct 138
Comenzile (cererile) limbajului de manipulare 140
Funcţia de creare standard şi comanda C (creare) 140
Comanda G (generare) . . . 143
Comanda S (ştergere) . . . 144
Comanda SE 146
Comanda M (atribuire) . . 247
Utilizarea funcţiei D (Denomber) 152
Utilizarea funcţiei NUMDE 153
Utilizarea funcţiei SUIVANT DE 153
Expresii 155
Conversii 156
Manipularea şirurilor de caractere 158
Rezervarea şi formatarea bufferelor de lucru . . 159
Comanda I (editare) 162
Editare standard 162
Editare cu format 164
Comanda SI (dacă) 166
Comanda POUR 168
Comanda FAIRE 170
Comanda DEPUIS 171
Comanda D 173
Comanda BLOQUER / LIBERER 174
Comanda LIRE/ECRIRE articole din fişiere secvenţiale pe suport magnetic 174
Citirea unui terminal într-un buffer 177
Modificări ale LMD-SOCRATE în VI.6.R 178
Calificare prin DONT 178
Variabile de lucru Wi . . . 179
Accesul direct prin criteriu . 181
Definirea constantelor alfanumerice 182
Comanda M 183
Comanda I 184
Testarea condiţiilor de execuţie a instrucţiunilor LMD . . 185
Citirea/scrierea unui terminal/buffer într-un buffer /pe un terminal 187
Definirea temporară a unei caracteristici de tip formal . . 187
Sortare 188
Cereri de constituire a punctelor de control 189
Compilarea condiţională a programelor 190
Compilarea textelor programelor de cereri 190
Concluzii 192

5. ALTE COMPONENTE ALE SGBD-SOCRATE 193

Macrogeneratorul 193
Definirea şi utilizarea macroinstrucţiunilor 194
Programe precompilate 200
Programe executabile — IMT 204
Programe de control a drepturilor de acces . . 207
Programe IMT pentru determinarea unei intrări în dicţionar (hashing) . . . 208
Programe la dispoziţia utilizatorului 209
Modificări ale macrogeneratorului în versiunea VI.6.R . . 213
Editorul de texte 215
Comenzi la nivel de text (fişier) 218
Comenzi la nivel de linie . . 220
Interfaţa cu limbaje evoluate . . 226
Semnificaţia şi parametrii punctelor de intrare 227
Sintaxa de apel şi descriere a parametrilor punctelor de intrare în limbajul COBOL . . 232
Sintaxa de apel şi descriere a parametrilor punctelor de intrare în limbajul ASSIRIS . 233
Sintaxa de apel şi descrierea parametrilor punctelor de intrare în limbajul FORTRAN . 235
Obţinerea programelor în format executabil 237

6. UTILIZAREA FUNCŢIUNILOR SGBD-SOCRATE 238

Definirea unei proceduri generale de apel a modulelor SGBD-SOCRATE 239
Utilizarea comenzii OPTION . . 240
Limbajul de comandă 242
Limbajul de comandă în prelucrarea „batch processing" . . 242
Comanda SOC 243
Comanda EVAL 243
Comanda NOEV 244
Comanda ACTI 244
Comanda DACTI 245
Comanda SAVE 245

Comenzi necesare descrierii fişierelor secvenţiale pe suport magnetic 245
Comanda RUN 247
Comanda LOGOUT . . . 248
Limbajul de comandă în prelucrarea conversaţională . . . 248
Comenzi la dispoziţia operatorului central 249
Comenzi la dispoziţia utilizatorului 250
Generarea bazei de baze . . . 255
Introducerea parolelor de acces sistem 258
Macroinstrucţiuni sistem pentru acces la datele conţinute în baza de date 258
Modificarea parolelor sistem 259
Editarea datelor tehnice ataşate bazelor de date gestionate de o bază de baze şi a parolelor sistem . . . 259
Editarea datelor tehnice ataşate unei baze de date . 260
Adăugarea în conversaţional al unui utilizator al unei baze de date 260
Adăugarea în „batch processing" a unui utilizator al bazei de date 261
Ştergerea unui utilizator al unei baze de date 261
Modificarea datelor ataşate unui utilizator 261
Scoaterea unei baze de date din evidenţa unei baze de baze 261
Utilitare care funcţionează sub controlul modulului XXXGBAS 262
Comanda SYSRUN 262
Generarea unei baze de date 263
Comanda VOL 263
Comanda UTI 264
Comanda ORG 266
Comanda EOF 266
Comanda ALLOC 267
Generarea bazei de date BDDPERS 267
Formatarea parţială sau totală a bazei de date 268
Statistici parţiale sau totale . 268
Dump parţial sau total . . . 269
Reorganizarea bazei de date . 270
Salvarea parţială sau totală a spaţiilor bazei de date . 270
Restaurarea parţială sau totală a subspaţiilor bazei de date 271
Posibilităţi de refacere a incoerenţelor fizice 272
Implementarea unei baze pe disc în VI.6.R 272
Concluzii 274
7. INTEGRITATEA BAZELOR DE DATE 274
Punct de referinţă 275
Punct de control 276
Înregistrarea punctelor de control 276
Jurnalul A 277
Jurnalul B 278
Cererea de constituire a jurnalelor A şi B 280
Reluări 281
Utilitare care funcţionează sub controlul modulului XXXREST . 281
Iniţializarea unui fişier de securitate 282
Restaurare la un punct de reluare 282
Redefinirea punctelor de referinţă 284
Mesaje de eroare 284
Concluzii 285
Secţiunea III. Optimizări şi dezvoltări de SGBD-uri
8. OPTIMIZAREA STRUCTURILOR BAZELOR DE DATE ŞI A PROGRAMELOR DE EXPLOATARE 286
Optimizarea structurilor şi a programelor pentru baza de date gestionate cu SGBD-SOCRATE . . 290
Niveluri de organizare şi gestiune a memoriei 290
Organizarea spaţiului virtual şi a spaţiului real . . 291
Principiul proiecţiei spaţiului virtual pe spaţiul real şi invers 296
Imaginea memoriei virtuale 302
Optimizarea structurilor la nivelul spaţiului virtual . . . 307
Optimizarea structurilor la nivelul spaţiului real 307
Optimizarea structurii bazei de date utilizînd metoda frecvenţelor de apariţie a datelor asociate caracteristicilor . . . 308
Optimizarea timpilor de acces necesari regăsirii unei informaţii 310
Optimizarea structurii bazei de date 311

Algoritm de determinare a dimensiunii informaţiei de cadraj 311
Optimizarea programelor de prelucrare 313
Determinarea optimă a spaţiului dicţionar al unei baze de date 313
Tehnici de construire a programelor de serviciu 315
Controlul coerenţei 315
Caracteristici de tip ⟨inel⟩ 317
Caracteristici declarate chei de acces 321
Şiruri de biţi 322
Suprimarea realizărilor de entitate 323
Reprezentarea relaţiilor de tip „m—n" 324
Definirea unor resurse partajate în contextul păstrării reentranţei programelor utilizatorilor conversaţionali . . . 326
Utilizarea „semafoarelor" pentru rezolvarea accesului concurent 327
Definirea unor resurse globale pentru o bază de date 328
Generarea automată a programelor de întreţinere şi/sau exploatare a bazei de date 329
Consideraţii preliminare . . 329
Metode de generare automată 330
Construirea generatoarelor de programe specializate pe clase de programe 332
Formalizarea descrierii structurii conceptuale . . 332
Formalizarea descrierii identificatorilor 336
Principiul funcţionării produsului de generare automată 340
Proiectarea structurii bazelor de date în vederea exploatării simultane 342
Integrarea bazelor de date aflate în exploatare 342
Integrarea bazelor de date aflate în proiectare 345
Concluzii 346

9. DEZVOLTĂRI SOCRATE ŞI ALTE SGBD-URI 348

Introducere 348
Dezvoltări ECA AUTOMATION 348
Versiunea VI.6 349
Versiunea VI.7 351
Versiunea VI.8 354
Concluzii 356
Dezvoltările grupului SYSECA — Sistemul CLIO 356
Introducere 356
Nucleul CLIO 358
CLIO-DBN 359
CLIO-DD 359
CLIO-BDA 360
Celelalte componente CLIO . . 361
CLIO-DML4 361
CLIO-HLI 361
CLIO-MGF 361
CLIO-RWF 361
CLIO-SMF 361
CLIO-SMF 361
CLIO-QF 362
CLIO produs porabil 363
Dezvoltări distribuite 363
Noţiuni de baze de date distribuite 363
Variante de distribuire pentru SGBD SOCRATE . . . 365
Realizări de SGBD-uri pe minicalculatoare în ţara noastră . . 367
SGBD-ARGUS 368
SGBD-LEDA 369
Resurse şi performanţe ARGUS, LEDA, SOCRATE-MINI 371

Secţiunea IV. Aplicaţie complexă cu SGBD-SOCRATE pentru activitatea de personal

10. PROIECTAREA BAZEI DE DATE PENTRU ACTIVITATEA DE PERSONAL 375

Metodica de proiectare a unei baze de date 375
Obiectivul activităţii de personal 377
Determinarea sferei de cuprindere a activităţii de personal . . 378
Relaţiile interne şi externe ale compartimentului P.I.R. — Fluxul informaţional 378
Documentele primare folosite în activitatea de personal — intrările sistemului 381
Situaţiile de informare — raportare privind activitatea de personal — ieşirile sistemului . . . 381
Alegerea sistemului de gestiune — de ce SGBD SOCRATE ? . . 387
De ce bază de date pentru activitatea de PERSONAL ? 389
Proiectarea structurii conceptuale a bazei de date 389

11. IMPLEMENTAREA BAZEI DE DATE PERSONAL 393

Generarea bazei de date . . . 393

Întreţinerea bazei de baze . . . 400
Modificarea parolelor sistem 400
Generarea bazei de date . . 401
Editarea datelor tehnice . . . 402
Catalogarea procedurilor şi formatarea bibliotecii de proceduri a aplicaţiei 405
Generarea automată a programelor de creare şi întreţinere a realizărilor de tip NOMENCLATOR — 415
Compilarea structurii conceptuale a bazei de date 424
Catalogarea programelor pentru baza de date 433
Exploatarea şi întreţinerea bazei de date 446
Testarea coerenţei fizice a bazei de date 452
Testarea coerenţei logice . . . 453

ANEXE 455
Anexa 1 Imaginea memoriei virtuale a bazei de date BDDPERS (V.1.5) 455
Anexa 2 Procedura LINKCOBO (V.1.5, V.1.6.R) 456
Anexa 3 Procedură generalizată pentru refacerea legăturilor de tip ⟨referire cu inel⟩ . . . 458
Anexa 4 Principiul de funcţionare al unui program de tip „CAM" 463
Anexa 5 Exemplu prototip de generare (Generaţia 1-a, Versiunea 2-a) 464
Anexa 6 Un exemplu de statistică obţinută pe baza de date ADMITBD (BDDPERS reorganizată prin utilitarele de salvare-restaurare SOCRATE) 469
Anexa 7 Un exemplu de incoerenţă fizică semnalată la execuţia unei statistici 471
Anexa 8 Mesaje de eroare (V.1.5, V.1.6.R) 474

Teste (9—25) 482
Bibliografie 495

Teste

Cititorul este invitat ca, pe măsura parcurgerii lucrării, să-şi verifice cunoştinţele, încercînd să rezolve testele propuse, mai întîi fără a se uita la răspunsuri.

1. Se dă structura :

```
ENTITE 5000 LUCRATOR
DEBUT
    —
    —    —
    —
    JUD-N (3 15) (ALBA ARAD
    CLUJ IASI)
FIN ?
```

Ce greşeală sintactică s-a produs la scrierea caracteristicii listă de valori JUD-N ?

Răspuns :

Sistemul dă eroare ER.D05, adică : numărul de valori dat în lista de valori este mai mare decît numărul specificat.

2. Se dă structura unei baze utilizator în care apare caracteristica DATA-INC, scrisă astfel :

```
DATA-INC 300101 A 99 1231
```

Ce eroare s-a produs ?

Răspuns :

Eroare ER D06 : Lipseşte tipul caracteristicii în definirea ei.

Trebuie scrisă astfel :

```
DATA — INC DE 300101 A 991231
```

3. S-a definit structura :

```
DEBUT
ENTITE 5000  LUCRATOR DEBUT
.
ENTITE 3     VECHIME DEBUT
.
FIN
ENTITE 4 CALIFICATIV DEBUT
.
FIN
FIN FIN
```

Unde s-a produs eroarea ?

Răspuns :

Sistemul dă eroare ED.D07, adică lipseşte semnul ? de la sfîrşitul definirii structurii.

4. Se dă următoarea structură de FORMAL :

```
FORMAL PERSONAL DEBUT
        NUME MOT 20
        MARCA DILATE 5
        GR-SANG MOT 4
        .
        COD-STUD DE 100000
        A 999999
?
```

Ude s-a produs eroarea ?

Răspuns :

Caracteristica COD-STUD trebuie să fie de tip DILATE sau MOT, PACKE şi trebuie încheiată structura prin FIN.

5. S-a scris un program pentru încărcarea unei entităţi. Programul nu are nici o eroare de sintaxă, iar din punct de vedere logic este bun.

Ce se va întîmpla după rularea acestui program, în cazul în care nu urmează linii de date, dar urmează un alt program de încărcare, astfel :

```
% RUN FN : R
   Program de încărcare a ENTITĂTII STUDENTI
   .
   ?
%      RUN FN:R
   Program de încărcare a ENTITATII PROFESII
   .
   ?
```

Răspuns : Se va obţine următorul mesaj :

```
*** ERREUR *** CARTE % LUE PAR
    L'UTILISATEUR
```

adică, într-o astfel de situaţie prima linie citită este luată drept linie de date. Ea are în prima coloană % şi este o linie de comandă SOCRATE.

Observaţie : În acest moment lucrul este abandonat, chiar dacă mai urmează şi alte programe de încărcare (în acelaşi JOB).

6. Se dă următoarea structură :

```
ENTITE 1000 MUNCITORI
     DEBUT
     MARCA DE 1111 A 9999 AVEC CLE
     UNIQUE FIN
     NUME MOT
     .
  FIN
```

şi formalul corespunzător :

```
     FORMAL FMUNCITORI
        DEBUT
          MARCA     DILATE   4
          NUME      MOT      30
          .
          FIN ?
```

Se scrie un program de încărcare a entităţii MUNCITORI iar liniile de date vor fi de următoarea formă :

```
1111POPESCU DAN. . . .      |
2345VASILESCU PETRE . .     |
5621GEORGESCU ILIE  . .     |  1002 linii (a
2345NEAGOE DUMITRU. . .     |  patra nu a
. . . . . . . . . . . .     |  fost memo-
9345POP SIMION . . . . .    |  rată)
3167DUMITRU MARCEL. .
```

Întrebare : Ce greşeli vor fi sesizate în urma rulării programului de încărcare.
Răspuns : a) la citirea celei de a patra linii de date sistemul va sesiza o realizare cu o cheie ce a mai fost generată şi va imprima mesajul :
COD **** ERREUR CLE DEJA PRESENTE ;
b) începînd cu linia 9345 POP SIMION .... ? nu vor mai fi generate realizării întrucît au fost generate 1000 de realizări cîte au fost declarate : va apare mesajul :
MUNCITORI ***** ERREUR ENTITE SATUREE

7. După un program de încărcare a unei entităţi, program care nu are erori de sintaxă şi din punct de vedere logic este corect, urmează linii de date care nu au structura conform machetei acceptate de program. Toate au erori şi deci nu pot fi încărcate.
După acest program urmează alt program de încărcare astfel :

```
JOB                                   cartele de date (eronate)
cartele de comandă                  % RUN FN:R
program de încărcare                  Alt program de încărcare.
?
```

Întrebare : Ce se va întîmpla în acest caz ?
Răspuns : Liniile de date fiind eronate nu vor fi încărcate şi sistemul va căuta cel puţin o linie de date corectă. Negăsind el va citi şi linia de comandă :
% RUN FN : R
după care urmează mesajul
***ERREUR ***CARTE % LUE PAR L'UTILISATEUR
şi va abandona lucrul.

8. Fie următoarea structură :

```
ENTITE 10000 MUNCITORI
  ┌─DEBUT
  │  MARCA
  │  NUME
  │  ADRESA
  │      ┌─DEBUT
  │      │  COD-LOC
  │      │  DEN-LOC
  │      │  STRADA
  │      │  NUMAR
  │      └─FIN
  │  LOC-MUNCA
  │  .
  │  .
  └─FIN
```

Pentru editarea unei anumite situaţii se scrie un program în care una din instrucţiuni este de următoarea formă :

```
I(60) ADRESA DE X2
```

Întrebare : este corectă această instrucţiune ?
Răspuns : Caracteristica ADRESA este de tip bloc şi trebuie scrisă astfel :

```
I (60)   COD-LOC   DE ADRESA DE X2
I (65)   DEN-LOC   DE ADRESA DE X2
I (90)   STRADA    DE ADRESA DE X2
I (115)  NUMAR     DE ADRESA DE X2
ECRIRE
```

unde X2 trebuie definit înainte, de forma
D X2 = UN MUNCITORI
O astfel de editare este admisă numai în cazul utilizării instrucţiunii I fără format, astfel :
I ADRESA DE X2.

9. Se scrie următoarea instrucţiune :
I(10)' SITUATIA PLECĂRILOR IN STRĂINATATE' ECRIRE ?
Întrebare : Ce se va întîmpla la execuţia ei ?
Răspuns : Fiind un şir de caractere, lungimea lui maximă va trebui să fie de 30 caractere. Deci, se va trunchia ; va apare astfel :
SITUATIA PLECARILOR IN STRAINA

Testele continuă la paginile 482–494

Baze de date și SGBD-ul SOCRATE

1. BAZE DE DATE : DEFINIRE ȘI CARACTERISTICI

Ce este o bază de date ?

Scopul principal al unei baze de date* constă în stocarea datelor în vederea satisfacerii facile a cerințelor conducerii, utilizînd tehnica de calcul electronică. Deci, apare ca un sistem de înmagazinare, regăsire, actualizare și întreținere a datelor necesare procesului de fundamentare a deciziei. Figura 1.1 prezintă o viziune simplificată a unei baze de date, unde notațiile au următoarea semnificație :

Pi — programe de aplicație,

Cj — colecții de date,

BD — bază de date propriu-zisă,

SGBD — sistem de gestiune a bazei de date.

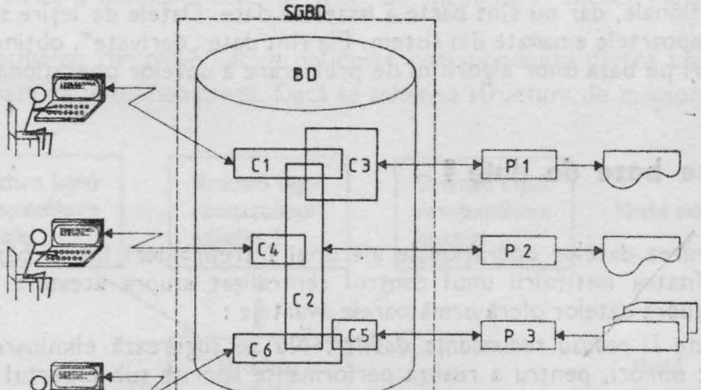


Fig. 1.1. Arhitectura simplificată a sistemului bazei de date.

* Menționăm că se folosește și noțiunea de „sistem de bază de date" în accepțiunea ce urmează.

Privită într-un sens larg baza de date implică patru componente : date, hardware, software şi utilizatori.

Datele sînt memorate pe purtători tehnici de informaţii.

Hardware-ul se compune din volume de memorie externă, pe care rezidă baza de date (discuri magnetice, tamburi magnetici etc.), împreună cu unităţile asociate (unităţi de discuri, canale etc.).

Software-ul asociat bazei de date, numit şi sistem de gestiune a bazei de date. Toate cererile de acces la baza de date sînt tratate de SGBD.

Utilizatorii bazei de date pot fi grupaţi în trei mari categorii :

- *utilizatorii finali* sînt cei ce interacţionează cu baza de date prin intermediul unui limbaj de interogare, sau care apelează programe scrise de programatorii de aplicaţii ;
- *programatorii de aplicaţii* sînt cei ce realizează programele de aplicaţii ale bazei de date, utilizînd limbajul de manipulare a datelor ;
- *administratorul bazei de date* este o persoană sau un grup de persoane responsabil cu controlul general al sistemului de bază de date.

Utilizînd noţiunea de date operaţionale se poate da următoarea definiţie a bazei de date [DATE 82] :

O bază de date este o colecţie de date operaţionale memorate, utilizate de sistemele de aplicaţii ale unei întreprinderi oarecare.

„Întreprindere" este un termen generic, folosit pentru orice organizaţie economică, comercială, tehnică, ştiinţifică sau de altă natură, de sine stătătoare. Orice întreprindere trebuie să menţină o mulţime de date necesare desfăşurării activităţii sale. Acestea sînt „datele ei operaţionale". Datele operaţionale nu trebuie să includă date de intrare sau ieşire sau orice alte date cu caracter tranzitoriu. Datele de intrare sînt datele care intră în sistem din lumea înconjurătoare. Ele pot schimba datele operaţionale, dar nu sînt parte a bazei de date. Datele de ieşire se referă la mesajele şi rapoartele emanate din sistem. Ele sînt date „derivate", obţinute de cele mai multe ori pe baza unor algoritmi de prelucrare a datelor operaţionale.

De ce baze de date ?

Memorarea datelor operaţionale ale unei întreprinderi într-o bază de date oferă posibilitatea instituirii unui control centralizat asupra acestora. Un astfel de control asupra datelor oferă următoarele avantaje :

— poate fi *redusă redundanţa datelor*. Nu se sugerează eliminarea întregii redundanţe ; uneori, pentru a realiza performanţe sporite sub aspectul vitezei de regăsire a datelor sau din alte considerente de ordin practic (datorate de cele mai multe ori caracteristicilor de implementare ale SGBD-ului), se acceptă un anumit grad de redundanţă. Într-o bază de date redundanţa poate fi controlată (spre deosebire de sistemele cu fişiere), SGBD-ului revenindu-i responsabilitatea propagării actualizărilor ; deci, în situaţia bazelor de date se vorbeşte de o redundanţă minimă şi controlată a datelor.

— se poate *evita inconsistenţa datelor*. Acest avantaj este un corolar al punctului anterior. Cînd redundanţa este înlăturată nu pot apărea inconsistenţe, iar cînd

redundar a există este controlată şi sistemul asigură propagarea actualizării la fiecare copie a aceleiaşi date.

— datele pot fi *partajate*. Partajarea datelor trebuie înţeleasă nu numai sub aspectul asigurării accesului mai multor utilizatori la aceleaşi date ci şi sub aspectul că pot fi dezvoltate noi aplicaţii fără să se modifice structura bazei de date.

— se poate forţa *standardizarea*. Utilizînd baza de date, administratorul bazei de date poate asigura aplicarea standardelor în reprezentarea datelor. Aceste standarde sînt : standardele întreprinderii, ale echipamentelor de tehnică de calcul, ale ramurii, ale economiei naţionale, internaţionale etc.

— se pot aplica *restricţii de securitate a datelor*. Avînd o jurisdicţie completă asupra datelor operaţionale, administratorul bazei de date poate asigura că accesul la baza de date se face numai prin canale corespunzătoare. În acest sens se pot defini verificări de autorizare, care să fie executate oricînd se încearcă un acces la anumite date.

— poate fi menţinută *integritatea datelor* prin existenţa unor proceduri de validare sau a unor protocoale de control concurent precum şi a unor proceduri de refacere a bazei de date după incidente.

— pot fi *echilibrate cerinţele conflictuale*. Cunoscînd cerinţele de ansamblu ale întreprinderii, în opoziţie cu cerinţele fiecărui utilizator individual, administratorul bazei de date poate structura baza de date în aşa fel încît să satisfacă cerinţele tuturor utilizatorilor în condiţii de redundanţă minimă şi controlată a datelor, pe de o parte, iar pe de altă parte, pot fi definite o serie de criterii de regăsire care să permită un acces rapid pentru aplicaţiile mai importante.

Multe din avantajele de mai sus sînt complet evidente. Un aspect nu este aşa de evident, deşi este un obiectiv mai degrabă decît un avantaj, şi anume asigurarea *independenţei datelor*.

Independenţa datelor

Se spune că o aplicaţie este dependentă de date dacă este imposibil să schimbi structura de memorare a datelor (cum sînt înregistrate fizic datele) sau strategia de acces la date fără să se afecteze aplicaţia.

Într-o bază de date nu se doreşte ca aplicaţiile să fie dependente de date, cel puţin din următoarele motive :

— Diferite aplicaţii au nevoie de viziuni diferite ale aceloraşi date. De exemplu, o aplicaţie utilizează un cîmp de dată drept o dată zecimală în timp ce o altă aplicaţie utilizează acelaşi cîmp de dată ca fiind reprezentat în binar. Sistemul va asigura automat conversia între reprezentarea internă a acelei date şi reprezentarea necesară fiecărei aplicaţii.

— Administratorul bazei de date trebuie să aibă libertatea să schimbe structura de memorare sau strategia de acces, ca răspuns la cerinţele de schimbare (întreprinderea trebuie să-şi schimbe standardele, priorităţile aplicaţiilor, unităţile de memorie etc.), fără să modifice aplicaţiile existente.

Independenţa datelor poate fi definită drept imunitatea programelor de aplicaţii la schimbarea structurii de memorare şi/sau a strategiei de acces.

Arhitectura unei baze de date

O arhitectură detaliată a unei baze de date în sens larg [DATE 82] este prezentată în figura 1.2.

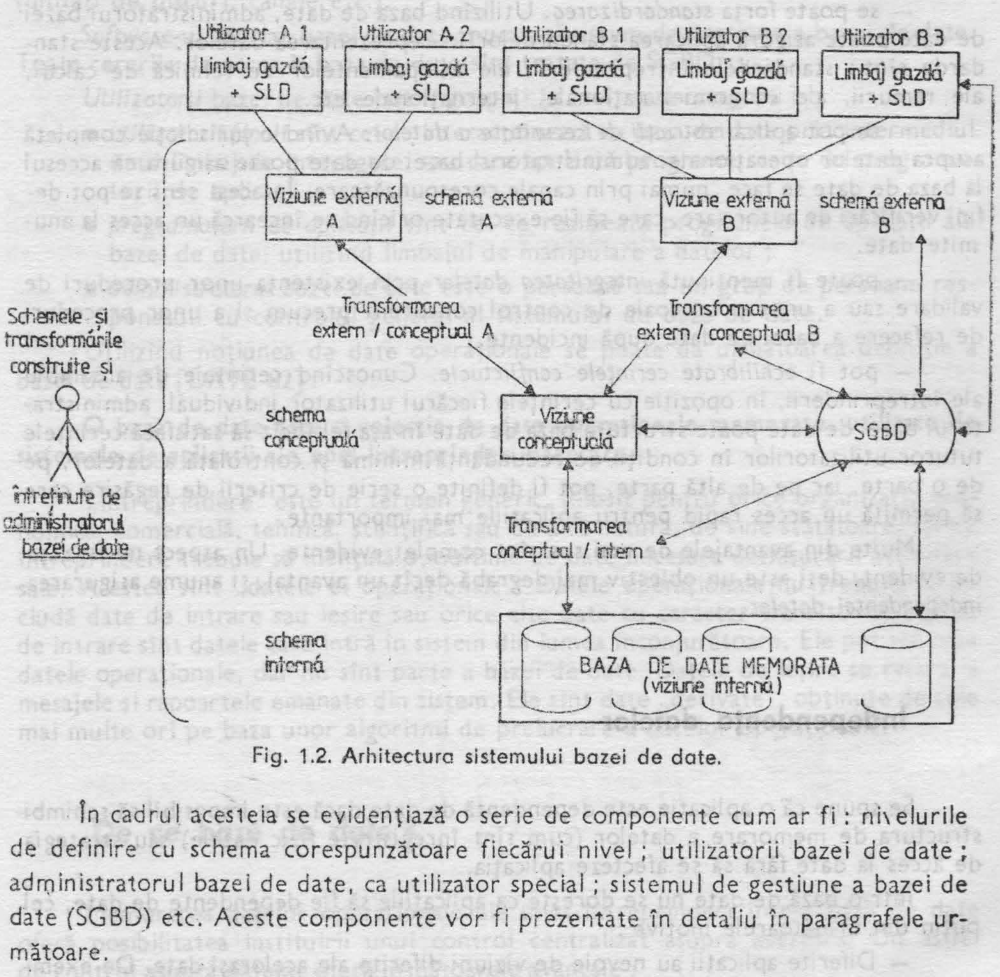


Fig. 1.2. Arhitectura sistemului bazei de date.

În cadrul acesteia se evidenţiază o serie de componente cum ar fi : nivelurile de definire cu schema corespunzătoare fiecărui nivel ; utilizatorii bazei de date ; administratorul bazei de date, ca utilizator special ; sistemul de gestiune a bazei de date (SGBD) etc. Aceste componente vor fi prezentate în detaliu în paragrafele următoare.

Nivele de structurare a datelor

Conform figurii 1.2 şi standardului ANSI/SPARC [JARDINE] pentru baze de date, datele pot fi definite pe trei niveluri : extern, conceptual şi intern (fig. 1.3). *Nivelul extern* descrie modul în care utilizatorii individuali văd baza de date. *Nivelul conceptual* descrie structura canonică a datelor operaţionale ale întreprinderii.

Nivelul intern descrie structura de memorare a datelor din baza de date şi strategia de acces la date.

O bază de date are mai multe viziuni externe, o singură viziune conceptuală şi o singură viziune internă, corespunzător celor trei niveluri de structurare a datelor.

Viziunea unui utilizator asupra bazei de date reprezintă o *viziune externă*. Fiecare viziune externă este definită prin intermediul *schemei externe*, utilizînd limbajul de descriere a datelor — LDD. Pot apărea diferenţe între definirile din schema externă şi cele din schema conceptuală (altă reprezentare, alt nume, altă ordine etc.). Este necesară, prin urmare, o transformare extern/conceptual.

Viziunea conceptuală este o reprezentare abstractă a tuturor informaţiilor din baza de date. Ea este definită prin intermediul *schemei conceptuale*, utilizînd LDD. Pentru a se obţine independenţa datelor schema conceptuală nu trebuie să conţină nici o referinţă la structura de memorare a datelor sau la strategia de acces. O astfel de definire se poate realiza într-o bază de date pentru personal, prin precizarea :

— tipului datelor elementare care specifică atributele obiectelor (exemple : marcă, nume, prenume, ...) :

— tipului datelor compuse care permit regruparea atributelor pentru a descrie entităţile lumii reale (exemple : persoana, data naşterii etc.) ;

— tipului datelor compuse care permit regruparea atributelor care descriu asocierile (relaţiile de agregare) entităţilor (relaţii de tip posesor-membru) ;

— restricţiilor de integritate a datelor (exemplu : anul naşterii să fie mai mare de 1950).

Viziunea internă, definită cu ajutorul schemei interne, descrie structura de memorare a datelor şi strategia de acces la ele (formatul intern al datelor, structura înregistrării fizice a datelor, secvenţa fizică de memorare, cheile, indexii şi pointerii etc.).

Transformarea conceptual/intern defineşte corespondenţa dintre viziunea conceptuală şi baza de date memorată. Dacă se schimbă structura de memorare a bazei

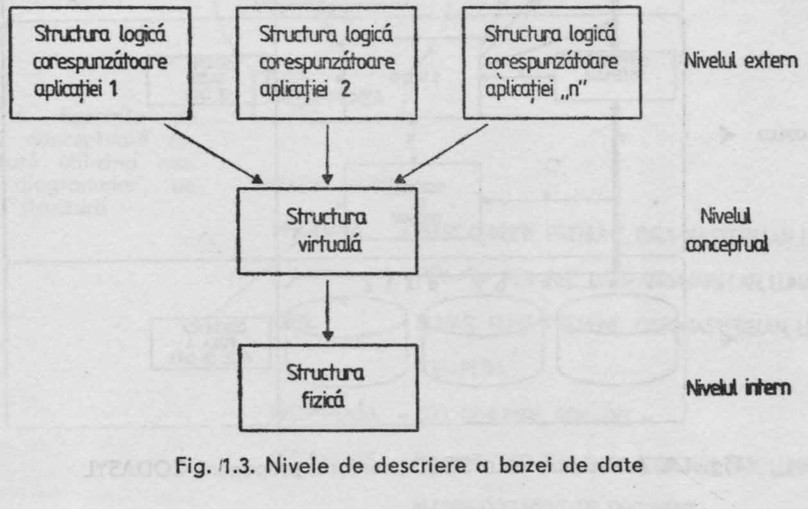


Fig. 1.3. Nivele de descriere a bazei de date

de date se va schimba corespunzător această transformare, astfel încît schema conceptuală şi aplicaţiile să rămînă neschimbate. Această structurare a bazei de date pe trei nivele : extern, conceptual şi intern, introdusă de ANSI/SPARC [STEEL, JARDINE], este cea care asigură independenţa datelor.

Sistemul de Gestiune a Bazei de Date (SGBD)

SGBD reprezintă software-ul care asigură realizarea următoarelor *funcţii* : definirea structurii bazei de date, încărcarea bazei de date, accesul la date (interogare, ştergere, modificare şi inversare), întreţinerea bazei de date (colectarea şi refolosirea spaţiilor goale, refacerea bazei de date în caz de incident), reorganizarea bazei de date (restructurarea datelor şi modificarea strategiei de acces), securitatea datelor.

În figura 1.4 se prezintă arhitectura unui SGBD în concepţia CODASYL. Din fig. 1.4 se pot deduce cele trei niveluri de organizare a datelor şi structurile corespunzătoare, precum şi operaţiile declanşate de derularea unui program de aplicaţie.

Succesiunea operaţiilor apare astfel :

1. Programul de aplicaţie A lansează o cerere de citire a unei date sau grup de date din bază. Cererea este lansată către SGBD.

2. Sistemul de gestiune interpretează cererea consultînd SUBSCHEMA referitoare la programul aplicaţiei A.

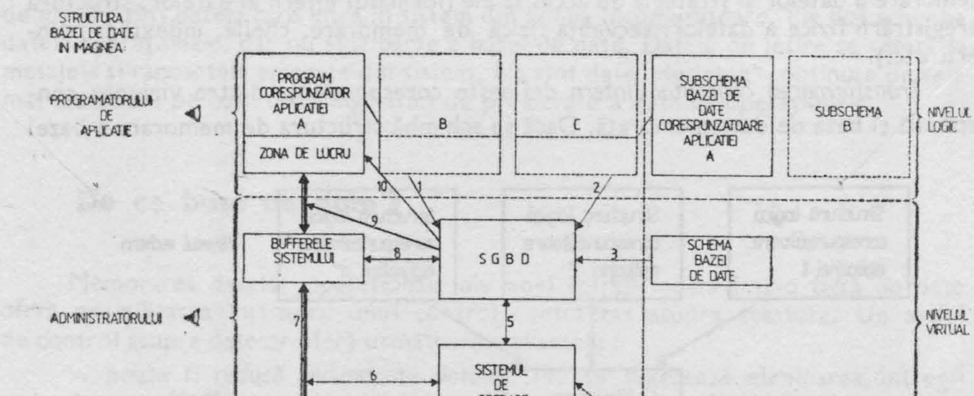


Fig. 1.4. Arhitectura unui SGBD conform propunerii CODASYL

3. Sistemul de gestiune apelează SCHEMA bazei de date şi determină la nivel logic datele solicitate.

4. Sistemul examinează descrierea fizică a bazei în raport cu cererea logică şi determină înregistrarea fizică ce prezintă interes.

5. Sistemul de gestiune lansează o comandă către sistemul de operare sub controlul căruia lucrează pentru căutarea înregistrării fizice de citit.

6. Sistemul de operare caută înregistrarea fizică.

7. Înregistrarea fizică găsită este transferată în memoria tampon a sistemului de gestiune.

8. SGBD procedează la o comparare între SCHEMA bazei de date şi SUB-SCHEMA corespunzătoare aplicaţiei A şi identifică datele solicitate de programul A.

9. SGBD transferă datele din memoria tampon în zona de lucru rezervată programului de aplicaţie A.

10. Programul de aplicaţie A preia controlul asupra tratării datelor solicitate iar pe parcursul executării programului se realizează un schimb de informaţii cu SGBD referitoare la „starea programului" sau eventualele erori constatate.

Operaţiile de scriere în Baza fizică sînt tratate de către un procesor, similar, toate modificările sau adăugirile sînt în general precedate de o operaţie de citire.

Utilizatorii bazei de date

Utilizatorii bazei de date pot fi programatorii de aplicaţie, utilizatorii finali, o persoană sau un grup de persoane care are în atenţie proiectarea bazei de date şi întreţinerea acesteia.

*

* *

Grupul de cercetători de la ANSI/SPARC prezintă o altă arhitectură [STEEL, JARDINE] pentru o bază de date privită ca sistem. Această arhitectură pune accentul pe interfeţele dintre componentele sistemului şi pe interfeţele dintre componente şi diferitele categorii de utilizatori (roluri umane). Partea importantă a arhitecturii ANSI este prezentată în figura 1.5.

Se remarcă cele trei roluri umane în definirea schemelor. Persoana sau grupul de persoane care defineşte schema conceptuală a bazei de date. Schema conceptuală reprezintă „cel mai bun model" pentru întreprindere. Ea furnizează o viziune pe termen lung şi este baza pentru declaraţiile de securitate şi integritate, precum şi standardele impuse de întreprindere diferiţilor utilizatori.

O persoană sau un grup care descriu o schemă externă pentru o aplicaţie particulară. Într-o întreprindere există mai multe roluri de acest fel.

Administratorul bazei de date are responsabilitatea definirii schemei interne a bazei de date. Tot el asigură şi întreţinerea acesteia.

Schemele astfel definite sînt verificate de procesoarele corespunzătoare şi, dacă sînt validate, sînt memorate în dicţionarul datelor.

Declararea schemei conceptuale a bazei de date se realizează prin intermediul interfeţei 1. După compilare definirea conceptuală este memorată în cadrul dicţionarului datelor, prin interfaţa 2. Administratorul bazei de date şi administratorii aplicaţiilor iau cunoştinţă de schema conceptuală prin intermediul interfeţei 3.

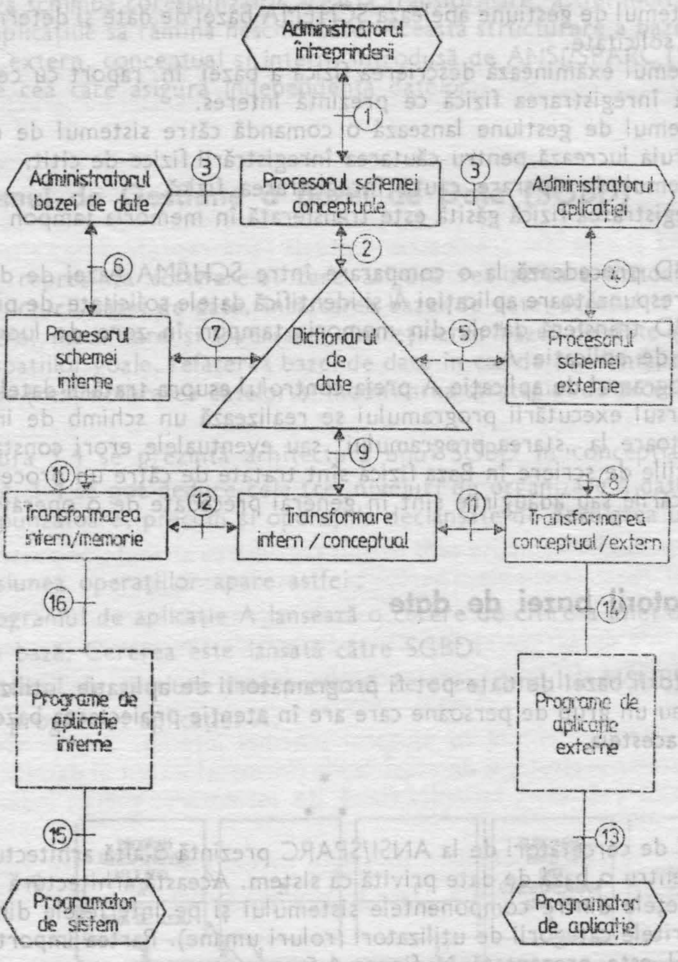


Fig. 1.5. Arhitectura ANSI a sistemului bazei de date.

Prin intermediul interfeţei 4 se specifică declaraţiile schemelor externe, a căror formă compilată este memorată în dicţionarul datelor prin interfaţa 5.

Declaraţiile schemei interne a bazei de date se specifică prin intermediul interfeţei 6 ; schema internă compilată este memorată în dicţionarul datelor prin intermediul interfeţei 7. Informaţiile din dicţionarul datelor sînt puse la dispoziţia modulelor de transformare prin interfeţele 8, 9 şi 10. Interfeţele 11 şi 12 sînt utilizate în timpul execuţiei pentru transmiterea datelor şi comenzilor între nivelele bazei de date (memorie, intern, conceptual şi extern).

Programatorii de aplicaţii şi utilizatorii finali comunică cu SGBD-ul prin intermediul limbajului de manipulare sau a limbajului de interogare (interfaţa 13). Prin intermediul interfeţei 14 schema externă este pusă la dispoziţia programului de aplicaţie pentru ca acesta să poată fi compilat/translatat/interpretat. La execuţie toate cererile de acces la date sînt transmise prin interfaţa 14 la sistemul de gestiune a bazei de date.

Dacă sînt necesare schimbări ale structurii de memorare sau a strategiei de acces (din motive de : performanţă, schimbarea suporţilor de memorare etc.) la date, programatorii de sistem prin intermediul interfeţei 15 specifică programele respective, al căror efect se transmite SGBD-ului prin intermediul interfeţei 16, fără a afecta nivelele conceptual şi extern.

Arhitectura ANSI pune, în acest fel, în evidenţă următoarele aspecte : definirea schemei conceptuale, definirea schemelor externe, definirea schemei interne, funcţiile de transformare şi independenta datelor.

Administratorul bazei de date (ABD)

ABD are următoarele responsabilităţi mai semnificative :

• decide conţinutul informaţiilor din baza de date (defineşte schema conceptuală). Aşa cum s-a arătat reprezentarea grafică a schemei conceptuale a bazei de date se poate realiza prin utilizarea a două modele :

1. *Modelul diagramelor de structură*, care utilizează două simboluri de bază pentru reprezentatea relaţiilor de tip 1 la n între entităţile de tip posesor-membru, astfel :

— dreptunghiul — prin care se reprezintă tipul datelor compuse, desemnate de acelaşi nume de entitate (această entitate este susceptibilă de a avea un număr oarecare de realizări). În interiorul dreptunghiului se trece numele entităţii ;

— săgeata — prin care se reprezintă relaţiile de agregare (asocierile) dintre realizările entităţii. La această reprezentare administratorul se preocupă de identificarea entităţilor şi a relaţiilor dintre ele. Reprezentarea atributelor se realizează într-un mod ales de administrator (figura 1.6).

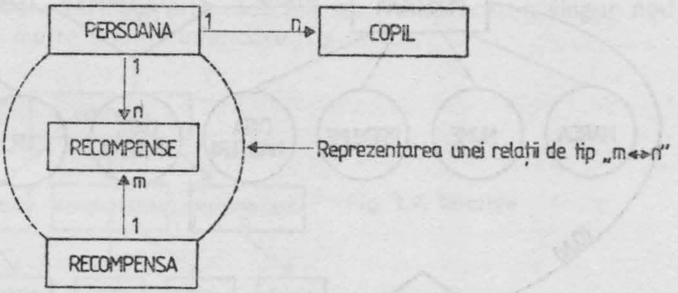


Fig. 1.6. Exemplu de schemă conceptuală reprezentată utilizînd modelul diagramelor de structură

Atributele entităţilor :

PERSOANA < MARCA, NUME, PRENUME, DATA-NASTERII (AN, LUNA, ZI) PER-COP, PER-REC, DATA-ANGAJARII (AN, LUNA, ZI) >

COPIL < NUMAR, NUME, PRENUME, DATA-NASTERII (AN, LUNA, ZI) COP-PER >

RECOMPENSA < COD, DENUMIRE, RCM-REC >

RECOMPENSE < REC-PER, REC-RCM, DATA-ACORDARII (AN, LUNA, ZI), NUMAR-DOCUMENT, TIP-DOCUMENT >

2. *Modelul entitate-asociere* care utilizează simbolurile :
— dreptunghiul — ca la modelul anterior ;
— rombul — pentru reprezentarea unui tip de asociere ;
— cercul — pentru reprezentarea unui atribut.

În cadrul acestor simboluri se trec identificatorii elementelor respective. Specificarea legăturilor existente între aceste simboluri se realizează cu ajutorul liniilor de conexiune. Pe aceste linii se poate specifica tipul asocierii. O astfel de reprezentare este prezentată în figura 1.7 ;

• decide structura de memorare şi strategia de acces, definind schema internă. Prin definirea structurii de memorare se stabileşte modul de descriere a stocării datelor pe suportul extern, care se poate referi la :

— fişierele care compun baza de date (nume, dimensiune, localizare, mod de organizare etc.) ;

— articolele care compun aceste fişiere (lungime, cîmpuri componente, mod de plasare etc.) ;

— căile de acces la articole (tabele de indexi, înlănţuiri, fişiere inverse etc..)

De exemplu, schema ilustrată în figura 1.6 poate fi reprezentată, utilizînd un tip de organizare dispersată a datelor, ca în figura 1.8 (numai entităţile PERSOANĂ şi COPIL). În acest exemplu realizările celor două entităţi au fost stocate în acelaşi fişier — FICH, iar toate tabelele de indexi în fişierul DICO. Indexul secundar utilizează un pointer spre următoarea intrare asociată unui nume de persoană, pentru care există intrări anterioare în dicţionar (acest index ar corespunde unei declaraţii de

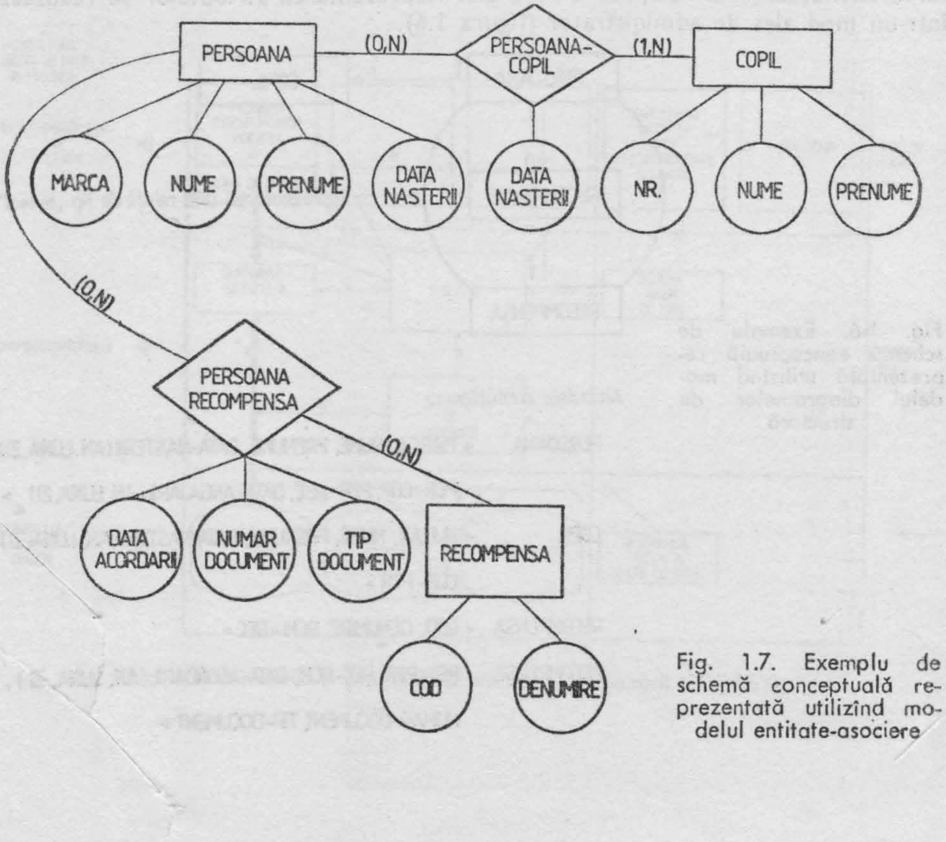


Fig. 1.7. Exemplu de schemă conceptuală reprezentată utilizînd modelul entitate-asociere

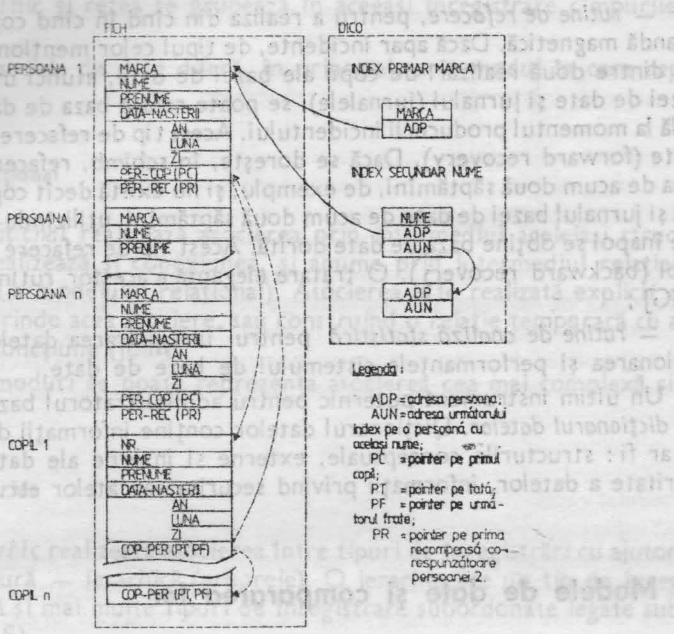


Fig. 1.8. Exemplu de schemă utilizînd organizarea dispersată a datelor

cheie care admite duplicate pentru valorile atribuite). Acest mod de organizare permite o regrupare a persoanelor care au acelaşi nume. Indexul primar pe MARCA arată faptul că se admit mai multe valori unice pentru acest index. Pentru a materializa relaţia existentă între cele două entităţi s-au utilizat pointeri care, de fapt, conţin adresele articolelor implicate. O viziune internă de acest tip, dar mai evoluată şi mai complicată realizează SGBD SOCRATE :

— stabileşte legăturile cu utilizatorii, definind schemele externe ;
— defineşte verificările de autorizare şi procedurile de validare ;
— defineşte strategia de refacere a bazei de date după incidente ;
— monitorizează performanţele şi realizează schimbările cerute pentru a mări performanţa.

Pentru îndeplinirea sarcinilor pe care le are ABD are nevoie de o serie de instrumente şi anume :

— *rutine de încărcare*, pentru crearea primei versiuni a bazei de date ;

— *rutine de reorganizare*, prin care se realizează colectarea şi eliminarea golurilor, rezultate în urma ştergerilor de înregistrări din baza de date. Tot în cadrul acestor rutine intră şi componentele care asigură reorganizarea bazei de date atunci cînd apar modificări în schema conceptuală ;

— *rutine de jurnalizare*, pentru refacerea bazei de date după incidente hardware sau software. Jurnalele reprezintă fişiere de memorie nedistructivă în care sînt înregistrate, în timpul lucrului cu baza de date, informaţii privind tranzacţiile de actualizare, operaţiile acestor tranzacţii asupra bazei de date, înregistrările pe care urmează să le modifice şi înregistrările modificate. Toate aceste informaţii sînt utilizate la refacerea bazei de date ;

— *rutine de refacere*, pentru a realiza din cînd în cînd copii ale bazei de date pe bandă magnetică. Dacă apar incidente, de tipul celor menţionate mai sus, în perioada dintre două realizări de copii ale bazei de date, atunci utilizînd ultima copie a bazei de date şi jurnalul (jurnalele), se poate reface baza de date în starea în care se află la momentul producerii incidentului. Acest tip de refacere se numeşte refacere înainte (forward recovery). Dacă se doreşte, în schimb, refacerea bazei de date în starea de acum două săptămîni, de exemplu, şi nu există decît copia de acum o săptămînă şi jurnalul bazei de date de acum două săptămîni, utilizîndu-le şi derulînd activităţile înapoi se obţine baza de date dorită. Acest tip de refacere se numeşte refacere înapoi (backward recovery). O tratare elegantă a acestor rutine poate fi găsită în [KING];

— *rutine de analiză statistică* pentru înregistrarea datelor statistice privind funcţionarea şi performanţele sistemului de baze de date.

Un ultim instrument puternic pentru administratorul bazei de date îl reprezintă *dicţionarul datelor*. Dicţionarul datelor conţine informaţii despre baza de date, cum ar fi: structurile conceptuale, externe şi interne ale datelor, restricţiile de integritate a datelor, informaţii privind securitatea datelor etc.

Modele de date şi compararea lor

Pentru a înţelege lumea reală este necesară plasarea unei anumite interpretări asupra obiectelor lumii reale. De exemplu, să decizi dacă anumite obiecte au o existenţă independentă sau există pentru a caracteriza alte obiecte. Un obiect poate fi sau independent (de exemplu, bloc) sau atribut ce caracterizează un obiect independent (de exemplu, adresa unui bloc). Pentru fiecare obiect atributele sale iau anumite valori. Valorile obiectelor, prin ele însele, nu au nici-o semnificaţie. Valoarea unei adrese (1 Mai, 742) sau a unei culori (verde) nu spun nimic în afară de faptul că există. Totuşi dacă se spune că: „blocul din b-dul 1 Mai nr. 742 este vopsit în verde", combinaţia celor două atribute transmite o anumită informaţie. Această informaţie este disponibilă datorită *conexiunii* stabilite între valorile celor două atribute ale obiectului independent, „bloc". O altă legătură, care pune în evidenţă informaţii, este *asocierea* dintre obiecte independente („proprietarul blocului din b-dul 1 Mai nr. 742 este întreprinderea IDEB").

Informaţiile dintr-o bază de date sînt reprezentate utilizînd ambele tipuri de legături. Reprezentările datelor din bazele de date se fac conform unor modele de date. În prezent sînt cunoscute trei modele de date:

— modelul relaţional;
— modelul ierarhic;
— modelul reţea.

(Nu luăm în considerare aici modelele semantice, modelele temporale, modelele bazate pe frame-uri etc., modele specifice inteligenţei artificiale, bazelor de date deductive, bazelor de date inteligente care nu fac obiectele acestei lucrări şi care nu au depăşit încă fazele de experiment de laborator).

Toate cele trei modele realizează în acelaşi fel conexiunea, şi anume grupînd în aceeaşi colecţie de date (obiect independent) cîmpurile de date (atributele acelui obiect). La modelul relaţional, relaţia grupează toate atributele care o caracterizează.

La modelele ierarhic şi reţea se grupează în aceeaşi înregistrare cîmpurile care o caracterizează.

Cele trei modele de date diferă, în principal, prin modul în care reprezintă asocierile.

Modelul relaţional

Modelul relaţional realizează asocierea prin intermediul aceleiaşi structuri de date prin care realizează şi conexiunea, şi anume prin intermediul relaţiei (unica structură de date a modelului relaţional). Asocierea este realizată explicit definind o relaţie care cuprinde acea asociere, sau construind o relaţie temporară cu ajutorul operatorului de joncţiune (join).

În ambele moduri se poate reprezenta asocierea cea mai complexă şi anume asocierea *n* : *m*.

Modelul ierarhic

Modelul ierarhic realizează asocierea între tipuri de înregistrări cu ajutorul unui nou tip de structură — ierarhică (arborele). O ierarhie are un tip de înregistrare definit ca rădăcină şi mai multe tipuri de înregistrare subordonate legate sub formă de arbore (fig. 1.9).

O bază de date care are la bază modelul ierarhic al datelor poate fi văzută ca o mulţime de arbori. Fiecare realizare a unei ierarhii are un nod rădăcină şi mai multe noduri subordonate.

Fiecare nod din arbore care nu este rădăcină sau frunză are un singur nod superior şi unul sau mai multe noduri inferioare (fig. 1.10).

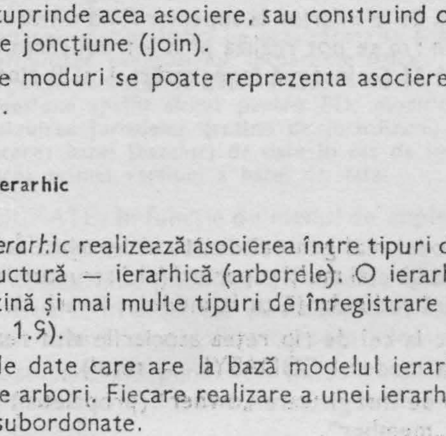


Fig. 1.9. Ierarhie

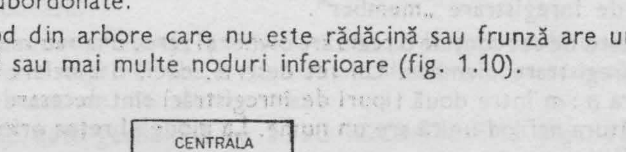


Fig. 1.10. Realizările unei ierarhii

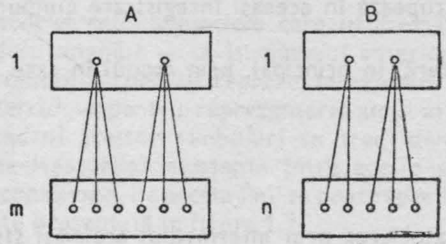


Fig. 1.11. Realizarea asocierilor n : m

Legătura de la superior la inferior este de *1 : n*, iar legătura de la inferior la superior este de *1 : 1*. Legătura de la inferior la superior fiind unică nu este numită, rămînînd anonimă. Asocierile *n : m* se pot realiza prin introducerea de redundanţe (prin familii de arbori) figura 1.11 (cele două înregistrări A şi B sînt memorate de două ori în baza de date).

Modelul reţea

Modelul reţea este un model mai general decît cel ierarhic. În cadrul acestui model fiecare inferior poate să aibă mai mulţi superiori (aici structurile de date de tip reţea sînt descrise direct şi fără redundanţă ca la modelul ierarhic).

Ca şi la modelul ierarhic la cel de tip reţea asocierile sînt realizate printr-o nouă structură — legătura (la modelul CODASYL — setul).

Un tip de set are un tip de înregistrare „owner" (proprietar) şi unul sau mai multe tipuri de înregistrare „member".

O realizare de set conţine o realizare owner şi zero, una sau mai multe realizări din fiecare înregistrare member. Un set descrie, deci, o asociere *1 : m*. Pentru a realiza legătura *n : m* între două tipuri de înregistrări sînt necesare două seturi. În acest caz, legătura nefiind unică are un nume. La modelul reţea orice tip de înregistrare poate fi :

— owner în orice număr de seturi ;

— member în orice număr de seturi ;

— member într-un set şi owner în alt set, dar nu poate fi owner şi member în cadrul aceluiaşi set (raportul DBTG din 1978 a încercat să ridice această restricţie).

Operatorii de manipulare din cadrul limbajelor de manipulare a datelor poartă amprenta modelului respectiv.

La *modelul relaţional* existînd o singură structură de date atît pentru conexiuni cît şi pentru asocieri este suficient cîte un operator pentru fiecare operaţie a bazei de date : înserare, ştergere, modificare, regăsire. Deci, uniformitatea reprezentării datelor conduce la o uniformitate corespunzătoare în mulţimea operatorilor. Modelul relaţional al datelor este, prin urmare, un model *simetric*.

La modelul *ierarhic* orice realizare de înregistrare are semnificaţie deplină numai cînd este văzută în contextul ierarhiei — nici-o realizare de înregistrare dependentă nu poate exista fără superiorii săi. La acest model dispare simetria operatorilor datorită structurii ierarhice asimetrice. Această asimetrie este principalul dezavantaj al modelului ierarhic, deoarece conduce la complicaţii pentru utilizatori. Referitor la operaţiile de actualizare apar unele anomalii :

— *Inserarea* — nu se pot introduce noi realizări ale unei înregistrări subordonate dacă nu sînt cunoscuţi superiorii.

— *Ştergerea* — dacă se şterge o realizare rădăcină a unei înregistrări, atunci se şterg automat toate realizările subordonate, pierzîndu-se astfel informaţii utile în continuare.

— *Actualizarea* — deoarece aceeaşi realizare a unui subordonat apare în mai mulţi arbori trebuie să decid fie că caut toate copiile unei date pentru a le actualiza, fie actualizez numai una din copii, introducînd în acest fel inconsistenţe în baza de date.

La modelul *reţea* dispar anomaliile de la modelul ierarhic (nu există redundanţe), dar şi aici sînt necesari mai mulţi operatori pentru a realiza aceeaşi operaţie:

— operatorul STORE pentru înserarea unei realizări de înregistrare şi
— operatorul CONNECT pentru stabilirea unei asocieri (legături).

Dezavantajul modelului reţea constă în complexitatea structurilor de date şi a limbajului de manipulare.

Analizînd cele trei modele de date se ajunge la următoarele concluzii:

— modelul ierarhic a fost dezvoltat acordîndu-se atenţie implementării bazei de date;
— modelul reţea a fost dezvoltat avîndu-se în vedere acordarea de facilităţi administratorului bazei de date şi utilizatorilor foarte competenţi;
— modelul relaţional a fost dezvoltat avîndu-se în vedere utilizatorii neinformaticieni.

Avantajele modelului relaţional faţă de celelalte modele de date

Modelul relaţional este un model de date simplu şi simetric. Aceste caracteristici au facilitat apariţia mai multor tipuri de limbaje relaţionale:

— limbaje care au la bază algebra relaţională,
— limbaje bazate pe calculul relaţional (tuplu şi domeniu) (QUEL, DEDUCE),
— limbaje bazate pe mapping (SQL),
— limbaje „fill in the blank" (QBE),

orientate pe diferite categorii de utilizatori şi aplicaţii, ceea ce nu s-a întîmplat cu celelalte modele de date.

Utilizarea sistemelor relaţionale este foarte simplă şi pentru nespecialişti prin faptul că:

— baza de date este văzută ca o colecţie de tabele (tabela este o structură comună oricărei persoane),
— toate sistemele au o interfaţă prietenoasă — User Friendly Interface (UFI) — care îl sprijină şi îl orientează pe utilizator în utilizarea sistemului,
— simplitatea limbajelor (număr redus de structuri şi operatori).

Modelul relaţional al datelor are la bază teoria matematică a relaţiilor. Aceasta a permis tratarea algoritmică a problemei proiectării bazei de date (problemă lăsată la dispoziţia utilizatorilor la celelalte modele) — aşa numita problemă a normalizării. Teoria normalizării a permis tratarea asistată de calculator a proiectării bazei de date. Normalizarea porneşte de la o mulţime de atribute (cîmpuri de date) şi o mulţime de dependenţe funcţionale între atribute şi obţine schema conceptuală a bazei de date relaţionale, cu relaţii într-o formă normalizată.

Această formă normalizată garantează faptul că anomaliile de actualizare nu vor mai apărea mai tîrziu cînd baza de date va fi încărcată, cînd vor exista multe programe

de aplicaţii pentru această bază de date şi deci nu va mai apărea necesitatea reproiectării bazei de date şi a modificării programelor de aplicaţii.

Teoria normalizării şi instrumentele care o implementează sînt foarte utile în etapa de proiectare a bazei de date şi „banalizează" oarecum sarcina administratorului bazei de date în această fază.

Care din cele trei modele va învinge? Greu de spus. Probabil toate trei la un loc. O bază de date va apărea relaţională pentru utilizatorii finali, reţea pentru administrator şi utilizatori pretenţioşi şi ierarhică pentru implementatorii ei.

Modelul relaţional aduce în plus teoria normalizării (proiectarea schemei conceptuale a bazei de date). Tehnica normalizării poate fi utilizată în proiectarea bazei de date chiar şi pentru bazele de date nerelaţionale (vezi [DRĂGHICI], [COJCCARU]). Au fost realizate chiar şi instrumentele software care utilizează această tehnică (la noi în ţară au fost realizate două astfel de instrumente, unul realizat în limbajul Pascal şi celălalt în limbajul Prolog — vezi [BARA]).

De ce tip este modelul datelor care fundamentează SGBD SOCRATE?

Pe baza celor enunţate SGBD SOCRATE se încadrează în clasa SGBD-urilor de tip reţea (asocierea dintre colecţii — entităţi — este realizată prin legătura — ANNEAU — REFERE; pentru realizarea structurilor de tip *n : m* nu introduce redundanţa datelor) dar are şi caracteristici ale modelului ierarhic (permite definirea de ierarhii prin imbricarea de entităţi).

Concluzii

Primul capitol al lucrării a realizat o prezentare a principalelor noţiuni din domeniul bazelor de date şi al sistemelor de gestiune a bazelor de date. Prezentarea făcută este teoretică — neavînd legătură cu o anumită implementare (în speţă cu SGBD SOCRATE). Conceptele prezentate au fost prelucrate din lucrări reprezentative pentru acest domeniu [DATE], [JARDINE], [STEEL], [ULLMAN]. Stăpînirea acestor concepte uşurează lectura şi înţelegerea lucrării. În capitolele următoare se va vedea modul în care SGBD SOCRATE respectă sau nu aceste „standarde".

2. SGBD SOCRATE – PREZENTARE SINTETICĂ

Definire şi componente

SOCRATE reprezintă un SGBD generalizat (utilizat pentru gestiunea bazelor de date cu structuri ierarhice sau în reţea, în orice domeniu de activitate) hibrid (are limbaje proprii şi interfaţă cu alte limbaje) [DU 82].

Permite exploatarea paralelă a mai multor baze de date atît în prelucrarea pe loturi cît şi în multiacces conversaţional (cu terminale sincrone sau asincrone) sau tranzacţional.

Componentele SGBD-SOCRATE sînt :

– limbajul de descriere a datelor (LDD) ;

– limbajul de manipulare a datelor (LMD) ;

– macrogeneratorul pentru gestiunea programelor scrise în LMD şi definirea unor limbaje utilizator specifice aplicaţiilor ;

– interfaţa cu limbajele COBOL, FORTRAN, ASSIRIS ;

– editor-corector de texte ;

– un set de utilitare pentru :

- definirea dicţionarului datelor ;
- definirea verificărilor de autorizare a accesului ;
- formatarea spaţiilor fizice alocate bazei de date conform specificaţiilor din dicţionarul datelor ;
- colectarea unor statistici pe baza cărora se pot determina performanţele ;
- determinarea eventualelor incoerenţe fizice ;
- reorganizarea bazei de date în cazul schimbării dicţionarului datelor (modificare dimensiune spaţiu alocat pentru BD, modificare dimensiune subpagină etc.) ;
- construirea jurnalelor (rutine de jurnalizare) ;
- refacerea bazei (bazelor) de date în caz de incident ;
- crearea primei versiuni a bazei de date.

SGBD-SOCRATE, în funcţie de modul de implementare (pe calculatoare Felix sau pe minicalculatoare), se prezintă astfel :

1) ca un nucleu unic (monitor de baze de date) în care funcţiunile sale sînt apelate ca procesoare. În acest caz este admisă utilizarea interactivă a funcţiunilor. Componentele utilizează funcţiunile incorporate în nucleu ca pe o resursă partajată ;

2) ca procesoare independente, îndeplinind în general o singură funcţiune. Utilizarea interactivă este posibilă numai prin intermediul sistemului de operare sub care funcţionează.

În acest caz componentele care permit accesul la baza de date sînt incorporate în fiecare funcţiune sau sînt puse la dispoziţia utilizatorului ca resursă partajată sub forma unor biblioteci de module reentrante.

O bază de date gestionată cu SGBD-SOCRATE este organizată pe suport magnetic cu acces direct (disc). Caracteristicile suportului extern, dimensiunea (dimensiunile) spaţiului (spaţiilor) alocate sînt caracteristici definite în dicţionarul datelor. Acest spaţiu este partajat (logic sau fizic) pentru a permite :

– memorarea dicţionarului bazei de date care reprezintă forma internă a structurii conceptuale a bazei de date descrise cu LDD ;

– memorarea datelor efective ;

– memorarea programelor (spaţiu gestionat de macrogenerator) ;

– formarea unor zone de lucru externe pentru procesoare.

Dacă spaţiul fizic este unic (un singur fişier în accepţiunea acestei noţiuni, pentru sistemul de calcul pe care este implementat) atunci va fi divizat logic în subspaţii, fiecare subspaţiu avînd destinaţii şi utilizări specifice. Dacă spaţiul nu este unic atunci fiecare spaţiu fizic, sau un ansamblu de spaţii fizice, este tratat ca diviziune logică.

În cazul utilizării unor spaţii fizice diferite se obţine o creştere importantă a caracteristicilor de exploatare pentru acel tip de implementare.

Spaţiul fizic alocat bazei de date poartă denumirea de *spaţiu real*. Indiferent care este modul de partajare (logică sau fizică) modul de organizare şi gestiune realizat de către SOCRATE este unitar.

Componentele SGBD-SOCRATE permit două moduri de utilizare a acestuia:

— *autonom* — în acest caz sistemul funcţionează ca un sistem închis care conţine toate elementele necesare utilizării sale (LDD, LMD, macrogenerator, editor-corector de texte);

— *metodă de acces* — în acest caz sistemul devine deschis pentru utilizarea sa din limbaje de nivel înalt.

Pentru ambele moduri de utilizare este admisă prelucrarea în:

— *„batch processing"* în care comenzile exprimate în limbajul de comandă SOCRATE şi cele exprimate în limbajul de comandă al uneia din componentele sale (LDD, LMD, macrogenerator, etc.) sînt stocate pe un fişier de comenzi. Structura şi modul de construire al acestui fişier de comenzi este dependentă de caracteristicile sistemului de operare gazdă. În general acest fişier de comenzi are structura cerută pentru intrarea standard a sistemului de calcul.

Caracteristic pentru acest gen de prelucrare este faptul că nu necesită intervenţia utilizatorului pe parcursul prelucrării. Această intervenţie a utilizatorului nu exclude dialogul normal, desfăşurat pe parcursul unei astfel de prelucrări, cerut de sistemul de operare gazdă;

— *„conversaţional"* în acest caz utilizatorul poartă un dialog cu sistemul SOCRATE de la un terminal conversaţional. Acest mod de prelucrare permite o programare şi exploatare interactivă.

Alegerea unui mod de prelucrare sau altul este dictată, în general, de caracteristicile aplicaţiilor funcţionale. Remarcăm faptul că programele de aplicaţie conversaţionale pot fi puse la punct si în prelucrarea „batch". În acest caz răspunsurile la dialog sînt preluate din fişierul de intrare al partiţiei în care se lucrează iar întrebările şi răspunsurile sînt afişate în fişierul de ieşire standard al partiţiei. Suportul fizic al acestor fişiere este dependent de caracteristicile sistemului de operare gazdă şi ale configuraţiei de calcul.

Istoric şi versiuni

Sistemul SOCRATE a cunoscut o dezvoltare în versiuni succesive şi a ajuns la implementarea pe o gamă largă de calculatoare.

Acest sistem este utilizat atît pentru gestiunea bazelor de date de aplicaţie cît şi ca software suport (pentru manipularea şi gestiunea datelor) pentru o serie de produse software generalizate (de exemplu SOGER realizat la CSP [S1], PACSIN realizat la ICSIT-TCI [P1]).

Un scurt istoric al dezvoltărilor succesive poate fi prezentat astfel:

— 1973 prima versiune programată în autocod IBM la Universitatea din GRENOBLE, Franţa, avînd la bază teza de doctorat a lui J. R. ABRIAL [S2];

— 1974 CII în colaborare cu ECA-AUTOMATION dezvoltă produsul pentru gama de calculatoare IRIS 45—80, 10070 şi SIEMENS sub numele SOCRATE V1.4.

Această versiune este prima versiune implementată la noi în ţară pe calculatoarele din gama FELIX;

— 1975 CII şi ECA-AUTOMATION dezvoltă versiunea V1.5.

Această versiune este ultima intrată la noi în ţară şi reprezintă software-ul de baze de date standard care se livrează împreună cu software-ul de bază al calculatoarelor din gama FELIX C-256—1024, FELIX C-5000.

În această versiune au fost cristalizate noţiunile de bază ale SGBD-SOCRATE, fiind element de referinţă pentru celelalte versiuni ;

– 1976, 1977, 1978 – ECA-UTOMATICN dezvoltă versiunile V.1.6, V.1.7 şi V.1.8 care vor fi prezentate în detaliu în capitolul 9.

Aceste versiuni au fost implementate pe calculatoarele din gama IRIS, BULL şi IBM.

V.1.6 a fost implementată şi pe minicalculatoare de tip SOLAR şi reprezintă SGBD-ul local pentru sistemul de gestiune a bazelor de date distribuite PLEXUS.

SOCRATE/II [DU 82] reprezintă un precursor al maşinilor de baze de date prin faptul că unele funcţiuni ale SGBD SOCRATE au fost cablate.

– SYSECA comercializează în prezent SGBD-SOCRATE sub numele de CLIO. Pentru acesta sînt oferite mai multe detalii în capitolul 9. De remarcat că în această formă SOCRATE funcţionează pe o varietate de calculatoare (micro, mini şi mari).

Dezvoltările efectuate asupra produsului în ţara noastră se referă la :

– V.1.6.R produsă de ICSIT-TCI care conţine dezvoltări proprii şi dezvoltări prevăzute în specificaţiile pentru V.1.6 şi V.1.7. Asupra modificărilor aduse de această versiune vom reveni ulterior în detaliu ;

– dezvoltări ale V.1.5. efectuate la C.C. al C.S.P. care se referă în principal la :

- exploatarea în condiţii de partajare a bazei de date în regim de multiprogramare ;
- efectuarea de calcule în dublă precizie ;
- utilizarea fişierelor pe disc magnetic ;
- utilizarea bibliotecilor IMT ca suport de introducere a programelor în format executabil ;
- definirea unui mecanism de tranzacţionare a operaţiilor de scriere în baza de date (inclusiv catalogare programe, adăugări la structură etc.) ;
- definirea rutinelor de reconstruire a bazei de date utilizînd jurnal de tip Q (quik).

V.1.5 şi V.1.6.R sînt operaţionale sub sistemele de operare gazdă SIRIS şi HELIOS.

– C.T.C.E. – CONSTANTA a realizat implementarea produsului pe calculatoarele din gama INDEPENDENT şi CORAL. Acest produs este operaţional sub sistemele de operare MINOS, MIX sau compatibile cu acestea (RSX 11/M). Este comercializat cu numele SOCRATE-MINI (MIDAS).

În această lucrare vom prezenta în detaliu versiunile existente în ţara noastră. V.1.5. va constitui baza acestor prezentări. Pentru această bază vom specifica difefenţele apărute în V.1.6 R (însăşi producătorul prezintă versiunea V 1.6.R ca diferenţă faţă de V 1.5).

Prin acest mod de prezentare am încercat să dăm volumului un caracter de instrument util atît în învăţarea principiilor SGBD-SOCRATE cît şi în utilizarea acestuia. Vom prezenta de asemenea caracteristicile de implementare pe minicalculatoare.

Atragem atenţia că majoritatea exemplelor prezentate (mai puţin programele IMT) conţin diverse „artificii" de programare valabile pe orice implementare deoarece sînt realizate ţinînd cont de filozofia generală a sistemului şi nu de o anumită implementare.

Mai mult exemplele prezentate pentru V.1.5 şi V.1.6.R descrise în LDD sau LMD pot fi testate pe Mini, substituind apelarea de procesor prin comanda „batch" sau $ GO x (în conversaţional) cu apelul procesorului respectiv al SOCRATE-MINI şi respectînd regulile de introducere a textului sursă ale acestuia.

Componentele principale

Limbajul de descriere a datelor.

Schema conceptuală a bazei de date, reprezentată grafic printr-unul din modelele descrise în § 1 este transpusă într-o formă acceptată de calculator (SGBD-SOCRATE) cu ajutorul limbajului de definire a datelor (LDD).

Elementul de descriere a datelor utilizat de LDD Socrate este caracteristică. Sînt permise următoarele tipuri de caracteristici:

— caracteristici elementare;
— grup de caracteristici (bloc);
— grup repetitiv de caracteristici (entitate);
— caracteristici de declarare a relaţiilor de agregare (inel şi referire);
— caracteristici inverse.

Fiecărei caracteristici i se acordă un identificator care trebuie să fie unic în ansamblul identificatorilor definiţi la acelaşi nivel într-un grup. Un nivel este definit printr-o declaraţie de bloc sau entitate.

De exemplu identificatorul AN poate fi definit şi în blocul DATA-NAŞTERII şi în blocul DATA-ANGAJĂRII aceste blocuri fiind atribute de tip grup ale aceleiaşi entităţi PERSOANA. Semantica acestui tip de obiect AN este dependentă de contextul în care apare an; pentru exemplul dat, ar fi an naştere şi, respectiv an angajare.

Caracteristicile elementare sînt de mai multe tipuri:

— ⟨*cuvînt*⟩, de exemplu: PRENUME MOT 25 în care PRENUME reprezintă identificatorul zonei, MOT este cuvîntul cheie care precizează tipul caracteristicii, iar 25 fixează lungimea zonei maxime rezervate;

— ⟨*listă de valori*⟩, de exemplu: APARTENENTA (4 5) (PCR, UTC, ODUS NM), unde APARTENENTA este identificatorul, 4 numărul de elemente din listă, 5 lungimea maximă a unui element (în caractere), iar PCR, UTC, ODUS şi NM sînt elemente din listă;

— ⟨*valoarea-numerică*⟩, de exemplu: AN DE 1921 A 2010, unde AN este identificatorul, DE precizează limita inferioară 1921 a intervalului iar A limita superioară a acestuia;

— ⟨*zecimal*⟩, de exemplu: VALOARE DECIMAL 15 V2, unde VALOARE este identificatorul, DECIMAL precizează tipul, 15 reprezintă numărul de cifre ale părţii întregi, V poziţia virgulei iar 2 numărul de cifre ale părţii fracţionare;

— ⟨*texte*⟩, de exemplu: CARACTERISTICI TEXTE 5, unde CARACTERISTICI reprezintă identificatorul, TEXTE precizează tipul iar 5 reprezintă numărul de linii pe care le rezervăm (a 60 caractere fiecare).

Acestor caracteristici, cu excepţia caracteristicii de tip text, li se poate asocia o declaraţie de acces. De exemplu, prin declaraţia de acces direct AVEC CLE UNIQUE FIN se specifică valori unice pentru realizările acelei caracteristici. Declaraţia de acces mai poate specifica modul de ordonare a valorilor (crescător/descrescător).

dacă se admit sau nu valori duplicate, dacă se folosesc algoritmii standard. Nu există restricţii asupra numărului de chei care se declară.

Caracteristicile descrise în baza de date pot fi grupate sub un identificator unic, printr-o declaraţie de tip bloc (introducerea unui nivel de calificare pentru un grup nerepetitiv) prin plasarea caracteristicilor între cuvintele cheie DEBUT (început) şi FIN (sfîrşit).

Exemplu :

```
DATA-NASTERII
 DEBUT
  AN DE 1980 A 2080 (2100)
  LUNA DE 1 A 12
  ZI   DE 1 A 31
 FIN
```

Este admisă definirea unui bloc în interiorul altui bloc. Acest tip de definire poartă denumirea de imbricare.

Entitatea reprezintă un bloc repetitiv, de exemplu ansamblul persoanelor unei întreprinderi se descrie :

Exemplu :

```
ENTITE 10000 PERSOANA
 DEBUT
  MARCA MOT 13 AVEC CLE UNIQUE FIN
  NUME MOT 25 AVEC CLE FIN
  PRENUME MOT 25
  DATA-NASTERII
   DEBUT
    AN DE 1980 A 2080 (2100)
    LUNA DE 1 A 12
    ZI   DE 1 A 31
   FIN
 FIN
```

unde ENTITE precizează tipul caracteristicii, 10000 reprezintă numărul maxim de realizări iar PERSOANA este identificatorul ansamblului.

Este permisă definirea unei entităţi în interiorul altei entităţi. Prin acest gen de definire LDD permite descrierea unui model ierarhic. În acest model ierarhic informaţiile sînt grupate pe niveluri, un nivel fiind introdus de o declaraţie de bloc sau entitate.

Deoarece structura unei baze de date este considerată un bloc, pe care convenim că se numeşte „FICHIER", atunci despre orice informaţie definită la acest nivel se spune că este definită la cel mai înalt nivel (numerotat convenţional cu 1).

Referindu-ne la exemplul precedent entitatea PERSOANA este la nivel 1 iar blocul DATA-NAŞTERII la nivel 2.

Între entităţile conţinute în baza de date se pot defini relaţii de tip posesor-membru prin intermediul caracteristicilor de agregare (ANNEAU-inel) şi (REFERE referire).

Caracteristica de tip inel (ANNEAU) permite introducerea unui potenţial posesor („owner") conform unei relaţii de tip 1-n iar caracteristica de tip referire (REFERE), asociată cu o declaraţie de tip inel, permite declararea unui potenţial membru („member").

De exemplu relaţia dintre PERSOANA şi COPIL (fig. 1.4, $ 1.6) poate fi materializată prin declaraţiile :

— PER-COP ANNEAU caracteristică situată în cadrul entităţii PERSOANA ;

— COP-PER REFERE PER-COP DE UN PERSOANA (caracteristică situată în cadrul entităţii COPIL).

Dacă o persoană are copii atunci caracteristica PER-COP va permite aflarea acestor copii iar COP-PER permite aflarea părintelui (relaţie 1-1).

Practic aceste caracteristici permit descrierea structurilor de date de tip reţea. O proprietate importantă a acestor declaraţii este aceea că exploatarea lor permite prelucrarea unor subdomenii disjuncte de date. Parcurgerea realizărilor grupate se face în ordinea secvenţială a definirii lor ca apartenenţă, ordinea de stocare putînd diferi de această organizare secvenţială.

Un alt mod de definire a unor subdomenii de realizări ale unei entităţi, este permis prin asocierea la o entitate a unor caracteristici de tip ⟨*invers*⟩ prin declaraţie, de exemplu :

BARBATI INVERSE TOUT PERSOANA, unde BARBATI este identificatorul, INVERSE precizează tipul (o construcţie similară unui fişier invers), TOUT este un cuantificator care spune că BARBATI se referă la toate realizările entităţii PERSOANA. Gruparea realizărilor entităţii asociate se va efectua conform unor *filtre logice* stabilite de administratorul bazei de date.

Important la aceste caracteristici este faptul că realizările grupate se parcurg în acces direct. Prin astfel de declaraţii baza de date poate fi înzestrată cu puternice mecanisme de acces. Numărul de caracteristici de acest tip asociate unei entităţi nu este limitat şi ceea ce este destul de important, aceste subdomenii nu trebuie să fie disjuncte.

Caracteristicile de tip entitate, referire cu inel şi invers definesc (subansambluri de tipuri de obiecte care se vor traduce, prin datele asociate, în (sub)ansambluri de date ale bazei de date.

Caracteristic acestui SGBD este faptul că toate tipurile de obiecte descrise în cadrul schemei conceptuale pot fi parcurse punctual sau grupat în acces secvenţial (accesul efectiv de date se realizează în mod direct).

Caracteristicile declarate chei pot fi parcurse, în funcţie de modul de declarare, în acces direct (CLE UNIQUE) sau secvenţial ordonat (AVEC CLE FIN) conform unei ordini crescătoare/descrescătoare de aranjare a valorilor.

Accesul direct este permis şi prin desemnarea, prin numărul său, a unei realizări de entitate, de exemplu PERSOANA 10. Caracteristica de tip invers asociată unei entităţi permite un acces direct similar, de exemplu BARBATI 10 va desemna tot PERSOANA 10 numai dacă aceasta satisface criteriile filtrului de grupare.

Toate tipurile de obiecte care descriu schema conceptuală vor fi plasate între cuvintele cheie DEBUT şi FIN. În acest mod SOCRATE consideră structura conceptuală ca un grup nerepetitiv de tipuri de obiecte (bloc). Utilizatorul poate atribui un nume acestui bloc sau nu (în acest caz numele va fi atribuit de sistem şi este „FICHIER").

Versiunile mai evoluate ale sistemului admit decuparea schemei conceptuale în subscheme. Această decupare se efectuează conform caracteristicilor deduse din modelul datelor. Important este faptul că baza de date permite atît o viziune integrală asupra acestora, ca şi cum ar forma un tot unitar, sau o viziune parţială ca şi cum ar fi independente. De exemplu, dacă baza de date reflectă datele operaţionale ale unei întreprinderi atunci un mod de descompunere ar fi cel funcţional : producţie, comercial, planificare-dezvoltare etc.

La definirea schemei de memorare a datelor administratorul poate aloca acestor elemente un spaţiu unic sau spaţii diferite.

Structura conceptuală, descrisă astfel cu LDD, devine structura logică a bazei de date sau mai simplu structura bazei de date.

Această structură conceptuală este analizată cu ajutorul compilatorului LDD. Dacă pe parcursul acestei analize nu se detectează erori care contravin regulilor de descriere atunci rezultatul compilării devine dicţionarul bazei de date. Acest dicţionar este conceput ca o *memorie virtuală* de o dimensiune foarte mare ($2^{31}-1$ cuvinte sau $2^{31}-1$ octeţi, dacă este organizată pe minicalculator) şi va fi utilizat pentru efectuarea transformărilor conceptual-intern sau conceptual-extern. Această memorie virtuală este denumită *spaţiul virtual* al bazei de date. Acest dicţionar conţine elementele necesare procesului de comprimare/decomprimare a datelor, informaţii referitoare la tabelele de indecşi asociate caracteristicilor declarate chei (număr de intrări alocate, algoritmul de gestiune etc.), condiţiile standard de validare a datelor etc. Acest spaţiu virtual alocat este paginat (divizat în pagini şi subpagini).

În acest spaţiu virtual fiecărui tip de obiect i se asigură o adresă, utilizînd metoda de alocare statică. Această adresă este unică pe parcursul întregii existenţe a obiectului respectiv.

Spaţiul disc asociat bazei de date este paginat (o pagină corespunzînd în general unei înregistrări fizice) şi poartă numele de *spaţiu real*. Aceste pagini sînt divizate în subpagini. În funcţie de implementare spaţiul asociat poate reprezenta, în sensul sistemului de operare gazdă, un singur fişier, caz în care este divizat logic în mai multe subspaţii, sau poate fi reprezentat de mai multe fişiere cu destinaţie strictă, în sensul divizării logice (Mini).

Dimensiunea unei subpagini (număr întreg de cuvinte) este aleasă de administrator în funcţie de proprietăţile datelor şi va reprezenta un parametru furnizat la definirea dicţionarului datelor.

În general spaţiul real este de o dimensiune mult inferioară celui virtual de aceea se realizează o proiecţie a spaţiului virtual pe spaţiul real, numai pentru subpaginile care conţin informaţie, prin intermediul unui mecanism de paginare. Acest mecanism de paginare simulează practic o alocare dinamică a spaţiului real.

Spaţiul real astfel divizat şi înzestrat cu un mecanism de paginare permite autoorganizarea (sînt alocate subpagini la cerere iar subpaginile care devin libere sînt plasate ca disponibile).

Principiul construirii memoriei virtuale, cel al paginării şi a modului de proiecţie a spaţiului virtual pe suportul real formează un tot unitar, pentru SOCRATE, reprezentînd o trăsătură caracteristică conservată de către toate versiunile şi implementările sale. De exemplu, dacă structura este decupată în subscheme atunci fiecăreia dintre acestea i se alocă o memorie virtuală maximă.

Dacă spaţiul disc alocat unei baze de date este unic atunci el va fi partajat prin împărţirea sa în sub-spaţii, în fiecare sub-spaţiu existînd acelaşi principiu de organizare şi gestiune.

Această viziune unitară a dus la apariţia ultimei versiuni SOCRATE (CLIO, cap. 9) care permite descrierea într-o bază de date *s* a structurii conceptuale căreia i se asociază *v* spaţii virtuale şi care se proiectează pe *r* spaţii reale.

Descrierea fişierelor externe bazei de date (secvenţiale, pe suport magnetic şi cartelă perforată, ecrane video etc.) se realizează cu ajutorul unor declaraţii speciale FORMAL sau FORMAT.

Aceste descrieri se asociază unor buffere de intrare/ieșire prin placarea descrierii lor ca o redefinire a acestora. Dimensiunea și structura acestor buffere este stabilită de utilizator. Dimensiunea maximă a unui buffer precum și numărul de buffere (minim 5) utilizabile simultan este o caracteristică de implementare. Aceste fișiere pot fi utilizate pentru a încărca și actualiza baza de date, a extrage informații în vederea unor alte genuri de prelucrări (eventuala comunicare a sistemului cu alte sisteme care au o altă tehnică de realizare) și pentru a forma ecranele terminalelor conversaționale. Utilizarea structurată a bufferelor nu este legată strict de execuția operațiilor de intrare-ieșire.

Aceste buffere pot forma un real instrument de lucru pentru diverse restructurări ale informațiilor conform algoritmului utilizatorului bazei de date.

Descrierea structurii din figura 1.6, utilizînd LDD SOCRATE, se prezintă astfel:

Exemplu:

```
DEBUT
 ENTITE 10000 PERSOANA
   DEBUT
     MARCA MOT 13 AVEC CLE UNIQUE FIN
     /* MARCA  REPREZINTA CHEIE PRIMARA;ACCEPTA VALORI UNICE*/
     NUME MOT  25 AVEC CLE ORDONNE FIN
     /* NUME  CHEIE SECUNDARA: ADMITE VALORI DUPLICATE */
     PRENUME MOT 25
     DATA-NASTERII                    /* BLOC                *
       DEBUT
         AN DE 1880 A 2080 (2100)
         LUNA DE 1 A 12 /* VALIDARE STANDARD:VALOAREA TREBUIE */
         ZI    DE 1 A 31 /* SA APARTINA INTERVALULUI DEFINIT      */
     FIN
     PER-COP ANNEAU /* COPIII PERSOANEI */
     PER-REC ANNEAU AVEC CHAINE DOUBLE FIN
     /* RECOMPENSELE UNEI PERSOANE INEL CU LANT DUBLU */
     /* EXPLOATABIL LIFO SAU FIFO                              */
     SOT REFERE UN PERSOANA
     /* DACA DATELE SOT(IE) SINT STOCATE IN ACEASTA BD */
     /* ATUNCI VOM DECLARA O  LEGATURA 1 — 1 PENTRU A MARCA */
     /* ACEST LUCRU                                               */
 FIN
 ENTITE 100000 COPIL
   DEBUT
     COP-PER REFERE PER-COP DE UN PERSOANA
     NR DE  1      A      15
     NUME MOT 25 /* PENTRU CAZUL CIND NUMELE NU COINCIDE CU */
                        /* AL TATĂLUI                              */
     PRENUME MOT 25
     SEX (2 1) (B F)
     DATA-NASTERII
       DEBUT
         AN DE 1984        A 2100
         LUNA DE 1         A 12
         ZI     DE 1        A 31
       FIN
   FIN
   ENTITE 100000 RECOMPENSE
     DEBUT
       REC-PER REFERE PER-REC DE UN PERSOANA
                   AVEC CHAINE DOUBLE FIN
```

```
REC-RCM REFERE RCM-REC DE UN RECOMPENSA
            AVEC CHAINE DOUBLE FIN
   DATA-ACORDARII
   DEBUT
      AN    DE 1944 A 21000
      LUNA DE  1   A 12
      ZI    DE  1   A 31
   FIN
  FIN
ENTITE 100 RECOMPENSA
  DEBUT
   RCM-REC ANNEAU AVEC CHAINE DOUBLE FIN
   COD DE 1 A 999 AVEC CLE UNIQUE FIN
   DENUMIRE
    DEBUT
      DENS MOT
      DENL1 MOT
      DENL2 MOT
      DENL3 MOT
   FIN
  FIN
  BARBATI INVERSE TOUT PERSOANA
  BAIETI INVERSE TOUT COPIL
FIN /* SFIRŞIT bloc structura */
?   /* sfirşit text definire şi lansare analiză sintactică */
```

Observaţii

Declararea tipurilor de obiecte care compun structura se efectuează în format liber, de la stinga la dreapta.

Singura restricţie impusă este că nu se admite despărţirea cuvintelor chele sau a indetificatorilor.

Pentru a face lizibilă descrierea sa administratorul poate utiliza o regulă de stil. Această regulă trebuie să pună în evidenţă, în principiu, apartenenţa (incluziunea structurală a atributelor).

Este permisă introducerea de comentarii prin plasarea lor între /* (inceput comentariu) şi */ (sfirşit comentariu).

/* şi */ asociat trebuie să apară pe aceeaşi linie sursă. În funcţie de implementare aceste comentarii pot apare oriunde între elementele care descriu un atribut, de exemplu, la V 1.5 şi V 1.6.R ; Mini nu admite despărţirea declaraţiei unui obiect cu comentarii. Declararea poate fi efectuată cu caractere mari, mici sau combinate.

Limbajul de manipulare a datelor

LMD permite utilizatorului să manipuleze datele stocate în baza de date invocînd elemente ale structurii sale descrise cu LDD. Această invocare nu necesită, comparativ cu limbajele clasice de nivel înalt (COBOL, FORTRAN, PL/I sau asamblare) (re)definirea formatelor logice ale datelor. Structura este pusă în factor pentru fiecare program de cereri sau cerere simplă exprimată în LMD.

LMD este un limbaj de nivel înalt cu sintaxa apropiată de limbajul natural (Franceză), cu o mare capacitate procedurală şi care posedă şi caracteristici neprocedurale.

Prin intermediul macrogeneratorului acest limbaj poate fi adaptat (prin traducerea cuvintelor rezervate) la un alt limbaj natural (română, de exemplu).

Acest limbaj pune la dispoziţia utilizatorului, pentru exprimarea algoritmilor săi, următoarele structuri de bază:

— structura secvenţială;

— structura alternativă;

— structura repetitivă.

Prin aceste structuri este forţată utilizarea metodei programării structurate la nivelul său cel mai înalt.

Un program scris în LMD (program de cereri) este un ansamblu structurat de cereri, terminat printr-un punct de interogare (?) O cerere simplă sau cerere este formată dintr-o comandă care determină acţiunea cerută şi eventual o frază care precizează obiectul (obiectele) asupra cărora se desfăşoară acţiunea. Obiectele acţiunii unei comenzi pot fi şi elemente externe structurii cum ar fi constantele.

Constantele care se pot utiliza în programele scrise în LMD pot fi de tipul:

— întregi, definite ca un şir de cifre zecimale precedate sau nu de semn (+ sau —);

Exemple:

+103; —1024; 175256;

— alfanumerice, definite ca un şir de caractere cuprinse între apostrofuri.

Exemple:

'ASE'; 'ICSIT-TCI'; 'SOCRATE V.1.5', „SOCRATE MINI";

— zecimale, definite cu o parte întreagă separată cu '.' de partea fracţionară şi precedate eventual de semn;

Exemple:

+3.14; 123.75; —09.8;

Constantele alfanumerice pot conţine caractere speciale introduse prin supraperforare sau prin codul lor în hexa (VI.6.R) conform unei construcţii speciale, explicată ulterior.

Acţiunile şi obiectele acţiunii unui program pot fi grupate astfel:

a) **Acţiuni asupra bazei de date**: aceste acţiuni sînt subordonate căutării unui element sau al unui (sub)ansamblu al bazei de date.

Desemnarea unui element determinat sau a unei caracteristici se realizează prin *citaţie*.

Evitarea ambiguităţii citaţiilor se realizează utilizînd:

— *calificarea* (ca în limbajul COBOL, de exemplu) prin care se desemnează filiaţia caracteristicii în conformitate cu structura arborescentă căreia îi aparţine. Această calificare poate fi realizată *postfixat*, utilizînd cuvîntul rezervat DE, sau *prefixat*, utilizînd cuvîntul rezervat POUR.

De exemplu, tipul de obiect AN definit în blocul DATA-NASTERII poate fi citat astfel:

comandă AN DE DATA-NASTERII, în notaţia postfixată, sau

POUR DATA-NASTERII *comandă* AN FIN, în notaţia prefixată.

Cele două forme de scriere sînt echivalente. Diferenţa constă în faptul că DE... este valabil numai la tipul de obiect pe care îl califică iar POUR...este valabil pentru toate tipurile de obiecte cuprinse între el şi FIN asociat. În acest mod POUR...

realizează o punere în factor a calificatorului şi implicit o grupare a cererilor pe care le conţine (grupează cererile în ansambluri de cereri). Pentru a realiza calificarea utilizatorul dispune de o varietate de modalităţi procedurale sau neprocedurale de exprimare a acesteia asupra cărora nu insistăm.

Observaţie

Declararea calificării prin cuvinte cheie este introdusă pentru a permite exprimarea în limbaj natural. Alte tipuri de SGBD-uri utilizează tehnici împrumutate din limbajele de programare. De exemplu, dacă DATA-NASTERII reprezintă o sub structură a structurii PERSOANA forma de calificare a elementului AN poate arăta astfel : PERSOANA.DATA-NASTERII.AN. În această construcţie pierderea calităţii exprimării naturale este evidentă.

Cuantificarea utilizînd, înaintea oricărei desemnări de ansamblu (entitate, inel sau invers), unul din cuantificatorii UN/UNE sau TOUT/TOUTE pentru a preciza dacă interesează unul sau respectiv toţi reprezentanţii ansamblului.

Exemplu :

1) comanda NUME DE UN PERSOANA arată că obiectul acţiunii este NUME care este atribut al entităţii PERSOANA, iar comanda se va efectua numai pentru prima (UN) realizare a entitaţii PERSOANA ;

2) comanda NUME DE TOUT PERSOANA arată că acţiunea se va desfăşura asupra caracteristicii NUME a tuturor realizălilor entităţii PERSOANA ;

– *selecţie* cu ajutorul filtrelor (AYANT, TELQUE).

Această selecţie permite efectuarea acţiunii comenzii numai asupra acelor elemente care îndeplinesc criteriile specificate în condiţia impusă de utilizator cu ajutorul filtrului.

Exemplu :

Dacă dorim să selectăm numai persoanele cu numele 'ION' din baza de date scriem astfel :

```
POUR TOUT PERSOANA AYANT NUME = 'ION' ;
    .comenzi
    .
    .
FIN
```

Condiţiile definite în cadrul unui filtru pot fi simple (introduse prin operatorii binari de comparare ca egal, diferit, mai mic, ...) sau compuse utilizînd operatorii logici ET (şi) şi OU (sau).

Exemplu :

Dacă dorim să imprimăm (I) MARCA tuturor persoanelor care au NUME egal cu 'ION' şi PRENUME diferit de 'CLAUDIA' scriem astfel :

```
POUR TOUT PERSOANA AYANT NUME = 'ION' ET
        PRENUME = 'CLAUDIA' ;
            I MARCA
FIN
```

Dacă nu există elemente care satisfac filtrul definit atunci nu se va desfăşura nici o acţiune precizată de comandă (comenzi).

b) **Acţiuni asupra variabilelor de lucru** : sistemul pune la dispoziţia fiecărui utilizator un set de variabile de lucru destinate, în funcţie de tip, unui anumit gen de operaţii.

Aceste variabile pot fi utilizate ca resurse globale pentru toate cererile pe care le execută un utilizator. Conţinutul lor poate fi transmis de la un program la altul ceea ce permite o înlănţuire parametrică a acestora. Mai mult, reprezintă o resursă globală utilizată de toate subrutinele unui program. Între program şi subrutinele sale există un mecanism mai evoluat de transmitere a parametrilor prin valoare.

Aceste variabile de lucru sînt de tipul:

1) *variabile de adresă* : X_i, i = 1, 9 (15 Mini) pentru lucru cu adrese virtuale. Aceste variabile permit marcarea unei realizări de entitate şi utilizarea lor drept calificator pentru a permite o exprimare mai comprimată a algoritmilor. De exemplu dacă citarea unui element al unei entităţi necesită o frază lungă pentru stabilirea filiaţiei sale această frază poate fi înlocuită cu marcatorul stabilit de către programator.

2) *variabile alfanumerice* : Z_i, i = 1,7 (10 Mini) pentru lucrul cu şiruri de caractere;

3) *variabile numerice* : Y_i, i = 1,25 (30 Mini) pentru lucrul cu numere întregi;

4) *variabile zecimale* : (V 1.6 şi Mini) : W_i, i = 1,5 (20 Mini) pentru lucrul cu valori zecimale.

Deoarece valorile numerice întregi sau zecimale sînt compactate în spaţiul real execuţia unor calcule cu aceste elemente se realizează prin transferarea lor prealabilă în variabile de lucru corespunzător tipului.

Exemplu:

Dacă vrem să obţinem vîrsta unei persoane (numai în ani) vom scrie:

M Y1 = AN DE DATA-NASTERII DE UN PERSOANA
M Y1 = 1987 — Y1 /* vîrsta în ani în Y1 */

Aceste variabile pot fi utilizate pentru operaţii de conversii (întreg-zecimal, zecimal-întreg, întreg-format extern, format extern-întreg etc.).

De exemplu prin M $Z_i = Y_i$ valoarea conţinută în Y_i este transformată într-un şir de caractere. Operaţia de conversie poate fi combinată cu concatenare/extragere şiruri de caractere.

De exemplu dacă marca conţine codul atribuit persoanei în cadrul „Sistemului de evidenţă a populaţiei" atunci va avea structura SAA. ..., unde S este secol-sex iar AA anul naşterii. Fără a ţine cont de valoarea lui S putem calcula vîrsta în ani a unei persoane.

Exemple:

M Z1 = MARCA DE UN PERSOANA
M Y1 = Z1 2 2 /* din poziţia 2 pe lungime 2*/
M Y1 = 87 — Y1 /* vîrsta în ani în Y1 */

Variabilele alfanumerice permit concatenarea, extragerea (sub)şirurilor de caractere.

c) **Acţiunile asupra programelor** se referă la:

1) *introducerea de cicluri* în prelucrări (inclusiv gruparea de cereri) prin: FAIRE...FIN, un ciclu poate fi reluat de la începutul său sau părăsit prin comenzile REFAIRE (reia), respectiv SORTIE (ieşi). Reluarea şi părăsirea pot fi efectuate în orice moment şi oriunde în program. Dacă nu avem nici un REFAIRE asociat la un FAIRE atunci construcţia va reprezenta numai un bloc de instrucţiuni.

2) *Introducerea unor cicluri de acces grupat* pe un (sub)ansamblu (entitate, inel sau invers) prin construcţii de forma:

POUR...FIN în care, în orice moment al prelucrării putem ieşi (SORTIE) sau putem să trecem la următorul reprezentant al (sub)ansamblului (SUIVANT). Dacă nu există o trecere forţată la alt reprezentant (SUIVANT) reluarea se face la întîlnirea comenzii FIN, asociate lui POUR. Citaţia care completează POUR poate

conţine unul sau mai multe filtre şi/sau poate preciza modul de acces ales prin simpla sa formă de exprimare. POUR îşi păstrează proprietatea de a califica prefixat tipurile de obiecte care sînt manipulate de comenzile pe care le regrupează.

Exemplu:

```
POUR TOUT PERSOANA AVEC MARCA⟩ '1';
     I NUME
FIN?
```

Se cere o parcurgere în acces direct a realizărilor şi selectarea acelora care au S (secol-sex)⟩1.

Acelaşi subansamblu poate fi parcurs secvenţial.

Exemplu:

```
POUR TOUT PERSOANA AYANT MARCA ⟩ '1';
     I NUME
FIN ?
```

Dacă se doreşte o parcurgere în ordine strict alfabetică a numelor se scrie astfel :

```
POUR TOUT PERSOANA PAR NUME :
     I MARCA I NUME /* MARCA şi NUME sînt calificate implicit */
FIN ?               /* prin PERSOANA */
```

3) *Introducerea de condiţii* : SI . . .FIN. Aceasta reprezintă structura alternativă IF-THEN-ELSE. Forma de exprimare este SI ⟨*condiţie*⟩ ALORS- - -[SINON . . .] FIN

Condiţia poate fi simplă, compusă sau test de existenţă. Prin existenţă se înţelege existenţa unei realizări de (sub)ansamblu sau o valoare diferită de zero binar (valoare nedefinită) atribuită unei variabile de lucru sau caracteristici elementare.

Un test de existenţă se introduce prin cuvintele cheie EXISTE (există) sau PAS (nu există).

De exemplu dacă există persoana 10 vrem să-i imprimăm (I) numele şi prenumele :

```
SI EXISTE UN PERSOANA 10
   ALORS
   I NUME I PRENUME
FIN ?
```

De exemplu dacă dorim să găsim persoanele din baza de date născute la data 21-03-1960 putem realiza acest lucru introducînd un filtru sau utilizînd o construcţie SI ...FIN, în care putem stabili o ordine de excludere a condiţiilor:

```
POUR TOUT PERSOANA                       Pentru fiecare persoană din toate
                                         persoanele existente
  SI AN DE DATA-NASTERII ]=1960          dacă are an de naştere diferit
                                         de 1960
    ALORS                                atunci
      SUIVANT                            treci la următoarea persoană
    SINON                                altfel
      SI LUNA DE DATA-NASTERII ]=3       dacă luna naşterii nu este 3
       ALORS                             atunci
         SUIVANT                         treci la următoarea persoană
       SINON                             altfel
         SI ZI DE DATA-NASTERII ]=21     dacă ziua de naştere nu este 21
           ALORS                         atunci
             SUIVANT                     treci la următoarea persoană
         FIN                             sfd; /* sfîrşit dacă */
      FIN                                sfd; /* sfîrşit dacă */
FIN                                      sfd; /* sfîrşit dacă */
I NUME I PRENUME /* scrie NUME şi */     scrie NUME scrie PRENUME
```

FIN; /* PRENUME pe terminal în conversaţional */ sfd; /* sfîrşit pentru */
/* sau pe ieşirea standard a sistemului */
/* în batch processing */

4) *Apel de subprograme*

Din punct de vedere al portabilităţii subprogramele pot fi :

— 100% portabile, reprezentate de programe scrise în LMD SOCRATE, care pot fi în format :

- *sursă*, sub formă de macroinstrucţiuni, care se apelează prin simpla citare a numelui atribuit, urmat eventual de o listă de parametrii ;
- *executabil*, sub formă de programe precompilate care se apelează cu comanda EXEC ⟨*nume*⟩.

Programele sursă *sînt compilate* în contextul programului apelant iar cele precompilate (compilate anterior utilizării) *se execută* în contextul programului apelant.

— 0%–100% portabile, reprezentate în general de programe în format direct executabil scrise în alt limbaj. Procentul de portabilitate este dat de procentul de portabilitate al limbajului respectiv.

Pentru V.1.5 şi V.1.6.R, aceste programe sînt scrise ASSIRIS iar pentru Mini în MACRO 11, COBOL, FORTRAN etc.

În versiunile V.1.5 şi V.1.6.R aceste programe se apelează ca si cele precompilate prin EXEC ⟨nume⟩.

La SOCRATE MINI apelul se face prin intermediul interfeţei cu limbaje evoluate, cu o construcţie de forma APPEL ⟨nume⟩ ... .

Subprogramele sînt gestionate de o componentă a SGBD-SOCRATE numită macrogenerator. Vom reveni asupra subprogramelor la tratarea acestei componente.

5) *Structurare pe niveluri*

Un nivel de structurare este definit de :

— un calificator (DE ... sau POUR...FIN) ;
— un cuantificator (UN/UNE ... sau TOUT/TOUTE ...) ;
— o condiţie (SI ... FIN) ;
— un grup de cereri ALORS (ALORS...SINON/ALORS...FIN) ;
— un grup de cereri SINON (SINON...FIN) ;
— o cerere FAIRE (FAIRE...FIN) ;
— un filtru : AYANT sau TELQUE...

Principalele comenzi ale LMD

O remarcă importantă asupra modului de acţiune asupra bazei de date a acestor comenzi este aceea că ele sînt foarte complexe. Deşi specializate pe tipuri de acţiuni aceste comenzi au o trăsătură comună : în funcţie de sensul acţiunii pot determina citirea şi/sau scrierea obiectelor pe care le manipulează. SGBD-SOCRATE nu are comenzi de citire/scriere în baza de date explicite. Accesul la baza de date se realizează prin mecanismul de paginare care are rolul de a optimiza accesul, de a citi datele asociate obiectelor acţiunii comenzilor, de a scrie eventualele pagini virtuale modificate şi de a recupera spaţiile reale eliberate prin operaţii de ştergere şi a le pune în lista de disponibilităţi.

Comenzile limbajului de manipulare a datelor SOCRATE sînt :

1) *Comanda de generare* G — permite crearea uneia sau mai multor realizări ale unui (sub)ansamblu.

Dacă acest (sub)ansamblu este entitate atunci rolul său se reduce la a marca faptul că spaţiul virtual asociat realizării este ocupat. Comanda poate acţiona asupra unei realizări bine precizate sau asupra primei realizări disponibile. Dacă realizarea desemnată există sau nu mai există realizări disponibile atunci semnalează incidentul produs.

Exemplu :

Dacă vrem să creem o realizare a entităţii PERSOANA atunci vom scrie : G UN PERSOANA.

Dacă dorim să creem toate realizările entităţii PERSOANA vom scrie G TOUT PERSOANA, sau dacă vrem să creem numai PERSOANA 10 vom scrie G UN PERSOANA 10.

Dacă (sub)ansamblul este o caracteristică de tip invers sau inel atunci comanda are o formă sintactică gen atribuire.

Exemplu :

Pentru a crea inversa BAIETI vom scrie
G TOUT BAIETI = TOUT COPIL AYANT SEX='B'.

Pentru a realiza legătura dintre COPIL 100 si PERSOANA 10 putem utiliza comanda G astfel :
G UN PER-COP DE UN PERSOANA 10 = UN COPIL 100

Pe astfel de caracteristici (inel) comanda G este interesantă cu utilizarea cuantificatorului TOUT. Pentru utilizarea punctuală există o comandă cu o formă mult mai concisă.

2) *omanda de ştergere* S – permite stergerea unei realizări de entitate cu eliberarea spaţiului virtual si real afectat. Dacă entitatea conţine entităţi imbricate atunci vor fi sterse şi realizările acestora. Eventualele caracteristici ANNEAU şi REFERE vor fi actualizate în mod automat (se păstrează coerenţa logică a acestora).

Exemplu :

Dacă dorim să ştergem PERSOANA 9 vom scrie :
S UN PERSOANA 9.

Dacă dorim să ştergem toate persoanele născute înainte de 1900 (inclusiv) scriem :
S TOUT PERSOANA AYANT AN DE DATA-NASTERII <= 1900 ;

3) *omanda de ştergere (sub)ansamblu (inel sau invers)* SE – permite suprimarea clasării într-un (sub)ansamblu.

Exemplu :

Dacă dorim să suprimăm clasarea efectuată în inversa BARBATI a realizărilor entităţii PERSOANA scriem :
SE TOUT BARBATI.

Exemplu :

Dacă dorim să eliminăm din aceeaşi clasare numai persoanele care nu au copii scriem :

SE TOUT BARBATI AYANT PAS PER-COP ;

Cererea SE UN PER-COP DE UN PERSOANA 10 va şterge legătura dintre cel mai tînăr copil (în sensul realizării legăturii) şi PERSOANA 10.

4) *Comanda de atribuire* M — permite efectuarea tuturor operaţiilor de atribuire utilizînd sintaxa generală

M ⟨stînga⟩ = ⟨dreapta⟩

Modul său de funcţionare este specific elementelor pe care le manipulează.

Exemplu :

Prin M Y1=Z1 se realizează conversia conţinutului variabilei Z1, reprezentat în zecimal extern, în binar.

Comanda M Y1 = ((Y3/2) * Y7 — 210) * (y5+3) + 1 atribuie lui Y1 valoarea calculată a expresiei aritmetice.

Comanda M NUME DE UN PERSOANA 10 = EXT va antrena un dialog standard de actualizare a caracteristicii NUME la terminal.

Dacă scriem M PRENUME DE UN PERSOANA 10 = 'CORINA' atunci constanta alfanumerică va fi atribuită pentru PRENUME.

În membrul drept al formatului de utilizare poate fi utilizată o funcţie ca :

— D — numără elementele unui (sub)ansamblu, de exemplu cu M Y1 = D TOUT BARBATI vom obţine în Y1 numărul bărbaţilor din baza de date ;

— NUMDE — permite aflarea numărului de realizare a unei realizări de entitate. Astfel cererea M Y5 = NUMDE UN PERSOANA atribuie lui Y5 numărul de realizare al primei persoane definite din baza de date.

Tot această comandă permite manipularea variabilelor de adresă X_i şi a funcţiilor pe adrese virtuale (SUIVANT).

De exemplu prin M X1 = UN PERSOANA 10

X1 va conţine adresa virtuală a persoanei 10 şi X1 va putea fi utilizat pentru a desemna această realizare. Prin M NUME DE X1=EXT se va antrena un dialog la terminal pentru actualizarea caracteristicii NUME a persoanei 10.

Această comandă este una din cele mai complexe comenzi a LMD. Am prezentat cîteva din acţiunile sale mai uşor de înţeles la acest moment. Acţiunile ei sînt prezentate în detaliu în § 4.

5) *Comanda* I — permite interogarea conţinutului caracteristicilor elementare sau a caracteristicilor grupate în (sub)ansambluri. Poate funcţiona într-un format standard de editare, format în care pentru fiecare caracteristică se imprimă numele urmat de : şi valoarea sa, sau format, în care numele caracteristicilor nu mai sînt imprimate ci numai conţinutul lor. În acest ultim caz imprimarea se face conform ordinelor de aranjare date de utilizator. Aceste ordine se specifică simplu prin precizarea poziţiei de început, a modului de aliniere şi eventual a lungimii pe care se face imprimarea. În acest mod utilizatorul poate redacta rapoarte de ieşire structurate conform necesităţilor sale.

Exemplu :

Dacă dorim să imprimăm datele asociate tuturor realizărilor entităţii PERSOANA scriem I TOUT PERSOANA. Dacă vrem să aflăm datele copiilor persoanei 10 scriem :

I TOUT PER-COP DE UN PERSOANA 10.

Caracteristicile sînt aranjate în pagină printr-un decalaj (idenţare) care sugerează incluziunea lor structurală.

Exemplu :

Dacă dorim să obţinem un tabel care să aibă forma liniei :

```
1        14              64        78
* marca  * nume şi prenume * data naşterii *
```

scriem :

```
POUR TOUT PERSOANA X1
   I (1) '*' I (+0 13) MARCA I(+0) '*'
   I (+0) NUME I (+1) PRENUME
   I (64) '*'
   I (+5 –4) AN DE DATA-NASTERII
   I (+3 –2) LUNA DE DATA-NASTERII
   I (+3 –2) ZI DE DATA-NASTERII
   I (78) '*' ECRIRE
FIN ?
```

Semnul „–" (minus) permite alinierea la dreapta a datelor de imprimat ; +0 specifică o operaţie de concatenare ; +5, +3, +1 specifică cu cîte caractere în dreapta faţă de poziţia la care s-a ajuns se face scrierea iar ECRIRE lansează scrierea efectivă.

Acest ordin de scriere (ECRIRE) este necesar deoarece formatarea se efectuează pe un buffer de maxim 120 caractere, iar scrierea efectivă trebuie efectuată numai cînd s-au introdus toate elementele desemnate de utilizator (deci un moment ales de acesta).

6) *Comanda* C – permite crearea unei realizări de entitate.

Exemplu :

C UN PERSOANA 10 ?

se antrenează un dialog la terminal cu utilizatorul prin care sînt cerute valori pentru caracteristicile entităţii PERSOANA.

În cazul nostru dialogul va fi (răspunsurile utilizatorului sînt subliniate, ⟨cr⟩ reprezintă apăsarea tastei retur de car) :

CREATION UN/UNE PERSOANA 10 ? **OUI** ⟨cr⟩ /* da */
MARCA : **1521001030011** ⟨cr⟩
NUME : **CREERE-STANDARD** ⟨cr⟩
PRENUME : ⟨cr⟩ /* PRENUME rămîne nemodificat */
CREATION DATA-NASTERII ? ⟨cr⟩ /* nu doresc creare */
SOT : **OUI** ⟨cr⟩ /* doresc realizarea legăturii cu soţ(ie) */
CITATION : **UN PERSOANA 5** ? ⟨cr⟩

Această comandă reprezintă o unealtă puternică pentru administratorul bazei de date necesară creării primei versiuni a bazei sale de date pe care poate să efectueze teste privind timpii de răspuns ai sistemului, de exemplu.

Comanda nu este încă disponibilă în implementarea pe Mini.

În afara acestor comenzi sistemul mai pune la dispoziţie comenzile : – LIRE/ECRIRE pentru citirea/scrierea articolelor din/în fişiere pe suport magnetic, citirea/scrierea liniilor de la/la terminal etc. O trecere în revistă a altor tipuri de comenzi a fost efectuată anterior la prezentarea acţiunilor asupra programelor.

Alte componente

Macrogeneratorul

Scopul macrogeneratorului este acela de a furniza o prelungire a limbajului de cereri, care modifică, pentru o aplicaţie dată, limbajul indus de structură şi înţeles efectiv de compilator.

Modul general de funcţionare al macrogeneratorului este acela de a înlocui şiruri de caractere în conformitate cu reguli (care pot fi introduse în orice moment) definite de utilizator şi de a conserva eventual aceste reguli.

În acest mod este permisă definirea unor limbaje utilizator oricît de specializate, apropiate de limbajul natural şi oricît de neprocedurale.

Regulile introduse de utilizator îi permit acestuia să definească o corespondenţă între un şir de caractere numit ⟨nume macro⟩ şi o suită de cereri, o cerere sau o parte bine determinată dintr-o cerere, care formează expansiunea macroinstrucţiunii.

Macrogeneratorul permite definirea de:

1) *Macroinstrucţiuni (macro)* care sînt scrise în LDD sau LMD. Aceste macroinstrucţiuni sînt stocate în spaţiul alocat bazei de date în format sursă. Pot avea un număr variabil de parametrii separaţi prin separatorii acceptaţi de compilatoarele LDD şi LMD sau fraze scrise într-un limbaj natural. Numele de macro împreună cu forma de definire a eventualelor parametrii formează modelul de apel al acestora. Parametrii care nu sînt definiţi în model se numesc parametri formali. Pentru gestiunea macroinstrucţiunilor dispunem de un set de ordine dintre care mai uzuale ar fi:

:DEFMAC — definire macroinstrucţiune;
:SUPMAC — suprimare macroinstrucţiune;
:EDIMAC — editarea macroinstrucţiunilor (în vederea prelucrării cu „editorul de texte");
:FDEF — anunţă sfîrşitul definirii.

Prin :DEFMAC se va specifica numele atribuit macroinstrucţiunii, modelul de apel şi modelul de expansiune care reprezintă frazele care vor fi substituite citării numelui macro.

Substituirea se realizează înlocuind, acolo unde apar, parametrii formali cu valorile acordate la apelul macro.

Exemplu:

Dacă dorim să traducem comanda SI ... FIN în echivalentul ei IF-THEN-ELSE scriem astfel:

```
:DEFMAC IF :EXP SI :FDEF?
:DEFMAC THEN :EXP ALORS :FDEF?
:DEFMAC ELSE :EXP SINON :FDEF?
:DEFMAC ENDIF; :EXP FIN :FDED?
```

:EXP defineşte corpul macroinstrucţiunii *(model de expansiune)*.

Utilizarea acestor macroinstrucţiuni se va efectua prin simpla citare a numelui lor. Pentru cazul nostru, de exemplu, putem avea următoarele citări:

```
IF MARCA > '1'
  THEN
   I NUME
```

```
ELSE
  I '*'
ENDIF;
```

Un alt exemplu, dacă dorim să introducem o frază a unui limbaj neprocedural care să arate astfel :

```
DORESC LISTA PERSOANELOR NASCUTE IN ANUL ...
ZIUA ... LUNA ... ?
```

Vom face declarațiile următoare :

```
:DEFMAC DORESC LISTA PERSOANELOR NASCUTE IN ANUL :
        ZIUA :    LUNA :
:EXP
 POUR TOUT PERSOANA X1
   AYANT
    AN DE DATA-NASTERII=:1:
    ET LUNA DE DATA-NASTERII=:3:
    ET ZI DE DATA-NASTERII    =:2: ;
    I (1) MARCA I (+1) NUME I (+1) PRENUME
    ECRIRE
 FIN
:FDEF ?
```

Prin :1:, :2:, :3: am citat parametrul formal cu numărul respectiv (numerotarea se face în ordinea de definire). Numărul lor poate diferi de la o implementare la alta (16-FELIX-C ; 30-Mini).

În acest caz se spune că parametrul este de tip „întreg" ; el va fi substituit, acolo unde apare, cu valoarea parametrului. Acest parametru poate fi un număr întreg, un identificator, o variabilă de lucru etc.

Acești parametri pot fi citați și sub forma ':n:', în acest caz fiind tratați ca un șir de caractere. Într-o macroinstrucțiune este admisă numai o singură formă de citare unui anumit parametru. De exemplu, dacă cităm parametrul 1 sub forma :1: nu mai avem voie să-l cităm și sub forma ':1:' un alt parametru poate fi citat sub această formă, de exemplu ':2:'.

Macroinstrucțiunea definită anterior poate fi apelată astfel :

```
— DORESC LISTA PERSOANELOR NASCUTE ÎN ANUL 1952
  ZIUA 1 LUNA 10 ?
— DORESC LISTA PERSOANELOR NASCUTE IN ANUL 1930
  ZIUA 23 LUNA 1 ?
```

Dacă dorim să apelăm acest program pentru un grup mai mare de date putem să-l introducem într-un ciclu, astfel :

```
FAIRE
    I 'ANUL' M Y1=EXT
    SI Y1=999 ALORS SORTIE FIN
    /* condiția de terminare ciclu */
    I 'ZIUA' M Y2=EXT
    I 'LUNA' M Y3=EXT
  DORESC LISTA PERSOANELOR NASCUTE
  ÎN ANUL Y1 ZIUA Y2 LUNA Y3
  REFAIRE /* reia ciclul */
FIN ?
```

Atribuirea valorilor lui Y1, Y2 și Y3 va fi solicitată la terminalul utilizatorului (sau la intrarea standard în „batch processing".

Utilizarea definirii macroinstrucțiunilor este interesantă și la efectuarea importului de programe SOCRATE din alte baze de date în sensul că se poate realiza o corespondență a identificatorilor utilizați de acestea. De exemplu, dacă un program

tratează un formal identificat cu numele MACHETA şi noi avem un formal identic, dar cu numele CARTELA nu mai este necesar şi importul definirii formalului MACHETA. Prin simpla declaraţie :DEFMAC MACHETA :EXP CARTELA :FDEF ? vom avea disponibil un formal „supranumit" MACHETA.

În modelul de expansiune este admis apelul altor macroinstrucţiuni şi a subprogramelor direct executabile.

Dacă dorim să suprimăm expansiunea (corpul) macroinstrucţiunii IF scriem astfel: SUPEXP IF ?

Dacă dorim să recuperăm modelul de expansiune asociat unei macroinstrucţiuni, de exemplu THEN scriem :EDIMAC THEN ?

2) *Programe precompilate*

Un program precompilat reprezintă o unitate direct executabilă şi eventual, adaptabilă la un context specific de execuţie. Un program precompilat nu are parametri formali. Eventuala transmitere de parametrii între un program precompilat şi apelantul său se realizează prin intermediul variabilelor de lucru, a zonelor descrise în structura bazei de date sau a bufferelor de lucru structurate conform necesităţilor utilizatorilor. Poate apela alte programe precompilate, macroinstrucţiuni sau programe direct executabile.

Pentru gestiunea programelor precompilate dispunem de un set de ordine, dintre care cele mai uzuale sînt:

:DEFPRO — pentru definirea programului;
:CONTXT — pentru definirea contextului de execuţie al acestuia;
:SUPPRO — pentru suprimarea unui program;
:EDIPRO — pentru editarea programului precompilat (obţinerea codului sursă corespondent).

Sursa unui program precompilat este analizată de către compilatorul LMD la catalogarea sa, iar codul rezultat este stocat, într-un format intern în spaţiul bazei de date.

Sub această formă devine utilizabil numai pentru baza de date respectivă.

Apelul său se realizează prin comanda EXEC ⟨nume⟩, unde ⟨nume⟩ este cel definit la catalogarea sa (:DEFPRO).

Programul care apelează macroinstrucţiunea DORESC ... în ciclu poate fi definit ca program precompilat.

Exemplu:

```
:DEFPRO PROG : CONTXT
:EXP  D  X1
 FAIRE I 'ANUL'
   M Y1=EXT /* VARIABILA Y1 VA CONTINE ANUL */
   SI Y1=999 ALORS SORTIE FIN /* 999 ESTE CONDITIA DE TERMINARE A CICLULUI */
   I 'ZIUA' M Y2=EXT /* VARIABILA Y2 VA CONTINE ZIUA */
   I 'LUNA' M Y3=EXT / * VARIABILA Y3 VA CONTINE LUNA */
   DORESC LISTA PERSOANELOR NASCUTE IN ANUL Y1
   ZIUA Y2 LUNA Y3
   REFAIRE
 FIN
:FDEF ?
```

Acest program se apelează prin EXEC PROG ?

3) *Programele în format executabil* sînt reprezentate de programe scrise în limbajul de asamblare al sistemului gazdă. Schimbul de parametri cu apelantul şi

forma de apel pot diferi de la o implementare la alta. Formatul general de apel este EXEC <nume>.

Aceste programe nu sînt în general portabile sau au un procent de portabilitate scăzut.

Cu acest gen de programe se poate acţiona direct pe formatul intern al bazei de date conform unor cerinţe speciale ale utilizatorului.

Editorul de texte

Editorul de texte este un instrument cu ajutorul căruia se pot efectua modificări asupra programelor/macroinstrucţiunilor sau a celor prezente curent în execuţie. Orice program devine, la momentul lansării sale în execuţie sau la momentul definirii sale, fişier curent al editorului de texte. Acest fişier curent devine disponibil pentru prelucrarea cu editorul de texte furnizînd, la cererea de introducere a unei noi prelucrări (QUESTION : în conversaţional), răspunsul $. Editorul de texte permite punerea facilă la punct a programelor, textelor de definire şi a caracteristicilor de tip <text> definite în baza de date.

Principalele funcţiuni ale editorului permit :

— crearea, listarea, punerea la zi a textelor ;
— inserarea de noi linii ;
— ştergerea sau înlocuirea unor linii ;
— substituirea unor şiruri de caractere la nivel de linie sau la nivelul întregului text ;
— gestionarea fişierelor de test ale definirii structurilor şi programelor cu ajutorul unui bibliotecar ;
— accesul la textele sursă ale macroinstrucţiunilor sau ale programelor precompilate.

Acest editor trebuie să funcţioneze atît prin apelul său explicit cît şi din celelalte componente.

Interacţiunea editor-componente (LMD, LDD, macrogenerator) permite o programare interactivă.

Implementarea pe minicalculatoare a SGBD SOCRATE nu dispune de un editor-corector de texte propriu. Sînt utilizate editoarele sistemului de operare (EDI EDT, ...). Interacţiunea între componente şi editor devine astfel externă, efectuîndu-se prin fişierele de text.

Interfaţa cu limbaje evoluate

SGBD SOCRATE pune la dispoziţia utilizatorilor săi o interfaţă cu baza de date pentru programele scrise în limbaje evoluate (COBOL, FORTRAN, PL/I, ASSIRIS, ...).

Acest lucru este realizat prin definirea unei metode de acces care constă în aceea că fiecare cerere de acces la baza de date exprimată în acest tip de programe se realizează prin intermediul unui modul de legătură, prevăzut cu un set de puncte de intrare funcţionale.

Metoda definită respectă principiul independenţei programelor faţă de date prin :

— definirea unor primitive de acces la baza de date ;
— simplitatea utilizării din limbaje evoluate ;

— modificările structurii bazei de date nu necesită recompilarea programelor scrise în limbaj evoluat deoarece modulul de legătură dă controlul unor subprograme scrise în LDD-SOCRATE (programe precompilate). În acest caz numai aceste programe trebuiesc recompilate şi/sau eventual, modificate.

Modul de apel al punctelor de intrare respectă un standard indiferent de limbaj iar efectuarea apelului se realizează conform sintaxei de apel a subprogramelor cerută de limbajul respectiv.

Principalele puncte de intrare şi funcţiunile îndeplinite sînt:

— SOPEN — deschiderea unei baze de date;
— SGBD — efectuarea accesului la baza de date (citire, scriere, interogare, modificare);
— SCLOSE — închiderea unei baze de date.

Dacă implementarea SGBD admite utilizarea jurnalelor „after" şi „before" pentru asigurarea integrităţii datelor sînt admise şi punctele de intrare:

— SSAVE — crearea unui punct de control;
— STERM — terminarea sesiunii de lucru şi închiderea fişierului jurnal.

Structuri de date acceptate de SGBD Socrate

Schema conceptuală a bazei de date poate reprezenta un anumit tip de structură de date sau o combinaţie de tipuri de structuri de date.

Pentru fiecare tip de structură de date vom preciza, în cele ce urmează, modul în care este descrisă în baza de date şi modul de reprezentare grafică în cazul schemei conceptuale. Vom prezenta în paralel reprezentarea prin modelul diagramelor de structură şi modelul entitate-asociere.

Structurile de date care pot fi modelate sînt:

1) *Structură punctuală*: această structură este reprezentată printr-o singură entitate, definită prin caracteristicile ei elementare. Realizările entităţii stocate în baza de date sînt independente faţă de celelalte entităţi din bază. Reprezentarea grafică a acestei structuri punctuale este prezentată în figurile 2.1 şi 2.2.

2) *Structură ierarhică lineară*: între entităţile descrise există relaţii de legătură lineare care pun în evidenţă relaţia posesor-membru. Entitatea posesor este numită „entitate tată" iar entitatea membru „entitate fiu". O „entitate tată" nu are decît o „entitate fiu".

Pentru exemplificare vom considera legătura existentă între persoană şi localitate şi legătura dintre localitate şi judeţul de care aparţine.

Acest exemplu se reprezintă grafic ca în figurile 2.3 şi 2.4.

Arborescenta indusă între realizările entităţilor JUDETE (J), LOCALITATI (L) şi PERSOANA (P) se poate reprezenta, schematic, ca în figura 2.5.

Această structură se interpretează astfel (pentru judeţul J1): judeţul J1 are n localităţi subordonate notate cu L_{11}, L_{12}, ..., L_{1n}. De localitatea 1 aparţin e persoane, de localitatea 2 aparţin p persoane etc.

3) *Structura ierarhică arborescentă*: o entitate este în relaţia 1 → n cu mai multe entităţi (o „entitate tată" are mai multe „entităţi fiu", iar o „entitate fiu" are o singură „entitate tată").

De exemplu, dacă dorim să reprezentăm pentru O PERSOANA relaţia existentă cu COPII săi şi calificativele obţinute în ultimii 10 ani vom putea reprezenta structura arborescentă ca în figura 2.6.

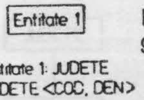


Fig. 2.1. Reprezentarea grafică a unei structuri punctuale (Bachman)

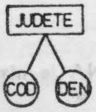
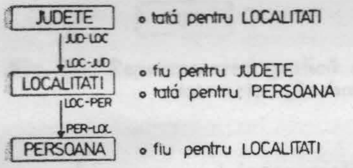


Fig. 2.2. Reprezentarea grafică a unei structuri punctuale (Chen)



JUDETE <COD, DEN, JUD-LOC>
LOCALITATI <COD, DEN, LOC-JUD, LOC-PER>
PERSOANA <MARCA, NUME, PREN, PER-LOC>

Fig. 2.3. Reprezentarea grafică a unei structuri ierarhice lineare (Bachman)

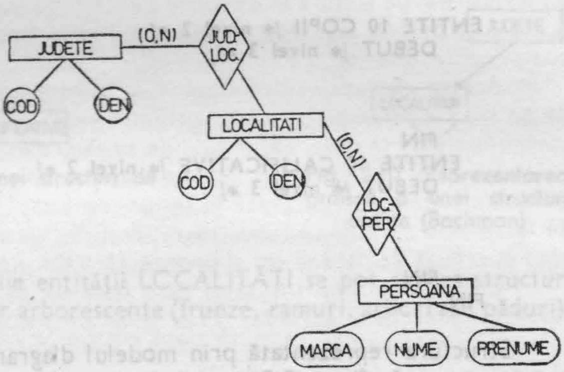


Fig. 2.4. Reprezentarea grafică a unei structuri ierarhice lineare (Chen)

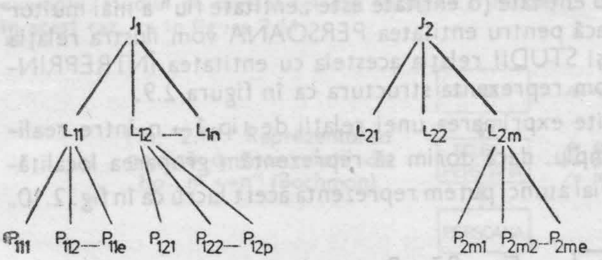
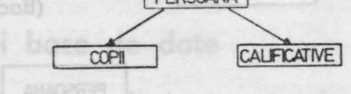


Fig. 2.5. Arborescența indusă între realizările entităților



PERSOANA <MARCA, NUME, PREN, PER-COP, PER-CAL>
COPII <NR, PREN, COP-PER>
CALIFICATIVE <AN, TIP, CAL-PER>

Fig. 2.6. Reprezentarea grafică a unei structuri ierarhice arborescente (Bachman)

În acest mod de reprezentare numărul de realizări ale entității COPII sau CALIFICATIVE poate fi oarecare.

Entitățile COPII și CALIFICATIVE pot fi definite ca entități imbricate în entitatea PERSOANA, dar în acest caz numărul lor maxim de realizări trebuie limitat (de exemplu 10).

Exemplu :

```
ENTITE — PERSOANA /* nivel 1 /*
DEBUT /* nivel 2 *

  ENTITE 10 COPII /* nivel 2 */
    DEBUT /* nivel 3 */
      .
      .
      .
    FIN
  ENTITE 10 CALIFICATIVE /* nivel 2 */
    DEBUT /* nivel 3 */
      .
      .
    FIN
FIN
```

Structura reprezentată prin modelul diagramelor de structură se reprezintă în acest caz ca în figura 2.7.

Reprezentarea prin modelul entitate-asociere a structurii din fig. 2.6 este prezentată în figura 2.8.

4) *structură reţea* : este o structură de tip multiposesor în sensul că mai multe entităţi sînt în relaţia 1 → n cu o entitate (o entitate este „entitate fiu" a mai multor „entităţi tată"). De exemplu, dacă pentru entitatea PERSOANA vom ilustra relaţia acesteia cu entităţile FUNCTII şi STUDII relaţia acesteia cu entitatea ÎNTREPRINDEREA legată de o localitate vom reprezenta structura ca în figura 2.9.

5) *structură ciclică* : permite exprimarea unei relaţii de tip 1 → n între realizările aceleiaşi entităţi. De exemplu, dacă dorim să reprezentăm gruparea localităţilor în sens administrativ teritorial atunci putem reprezenta acest lucru ca în fig. 2.10.

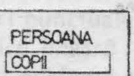


Fig. 2.7. Reprezentarea grafică a unei structuri ierarhice arborescente cu entităţi imbricate (Bachman)

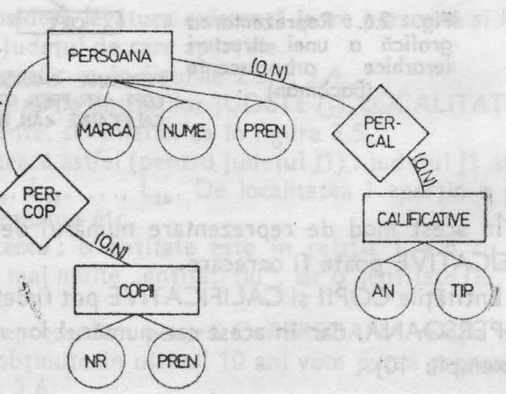


Fig. 2.8. Reprezentarea grafică a unei structuri ierarhice arborescente (Chen)

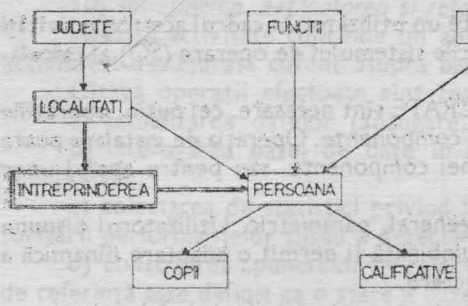


Fig. 2.9. Reprezentarea grafică a unei structuri de date rețea (Bachman)

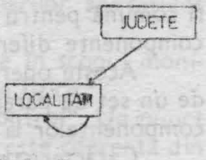


Fig. 2.10. Reprezentarea grafică a unei structuri ciclice (Bachman)

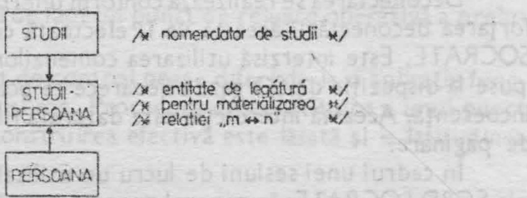
În acest mod între realizările entității LOCALITĂȚI se pot realiza structuri de date din toată gama structurilor arborescente (frunze, ramuri, arbori sau păduri).

Observație

Deși aceste structuri pun în evidență relațiile de tip „1 → n", este permisă și materializarea relațiilor de tip „m — n" conform unui artificiu de descompunere al acestora în relații elementare „1 — n".

De exemplu, dacă dorim să surprindem faptul că o persoană, are satisfăcute mai multe studii putem înlocui relația „1—1" dintre PERSOANA și STUDII (care dă în acest caz, de exemplu, studiul de nivelul cel mai înalt) cu o relație „1 → n". Schema conceptuală ar arăta, în acest caz, ca în figura 2.11.

Fig. 2.11. Reprezentarea grafică a unei relații de tip „m ↔ n" (Bachman)



Pentru acest tip de relații oferim mai multe amănunte în capitolul 10.

Ciclul de viață al unei baze de date

Principalele activități desfășurate de-a lungul ciclului de viață al unei baze de date, gestionată cu SGBD SOCRATE sînt:

1) restaurarea componentelor software ale SGBD;
2) generarea dicționarului datelor;
3) definirea parolelor sistem de acces la dicționarul datelor;
4) definirea caracteristicilor asociate spațiilor fizice alocate bazei de date;
5) definirea utilizatorilor și a drepturilor de acces ale acestora;
6) formatarea spațiilor alocate bazei de date;
7) compilarea schemei conceptuale a bazei de date;
8) testarea și catalogarea programelor necesare aplicațiilor bazei de date;
9) încărcarea bazei de date;
10) întreținerea, exploatarea și refacerea bazei de date;
11) reorganizarea bazei de date.

Operaţiile pe care trebuie să le execute un utilizator în cadrul acestor activităţi sînt dependente, în general, de caracteristicile sistemului de operare (SO) al calculatorului gazdă.

În cazul lucrului cu componentele SOCRATE sînt necesare, cel puţin, operaţiile de instalare şi de lansare în execuţie a acelei componente. Operaţia de instalare poate fi comună pentru mai multe apeluri ale unei componente sau pentru apelul unor componente diferite.

Aceste operaţiuni sînt efectuate, în general, parametric. Utilizatorul dispune de un set de parametrii a căror utilizare combinată îi permit o adaptare dinamică a componentelor la necesităţile sale de prelucrare.

Caracteristic pentru utilizarea acestor componente este faptul că interacţiunea utilizator-sistem se efectuează în sesiuni de lucru. Pentru un utilizator dat o sesiune de lucru este definită ca suma operaţiilor efectuate de acesta între momentul conectării la o bază de date şi cel al deconectării de la acea bază de date.

Conectarea unui utilizator la o bază de date se face conform unui protocol stabilit în *procedura* LOGIN. Prin acest protocol sînt cerute utilizatorului elementele necesare identificării bazei de date (sau a porţiunii din baza de date) cu care doreşte să lucreze şi datele sale personale necesare procesului de autentificare (nume, parola, cont etc.). Dacă aceste elemente sînt corecte, din punct de vedere al verificărilor impuse de procedura de confirmare, utilizatorului i se acordă dreptul de acces la baza de date.

Pe parcursul lucrului se verifică permanent dreptul de acces al utilizatorului la operaţiile (comenzile) pe care doreşte să le efectueze.

Dacă operaţia cerută contravine drepturilor sale de acces atunci ea va fi refuzată.

Deconectarea se realizează conform unei *proceduri de* LOGOUT. Dacă se doreşte forţarea deconectării aceasta va fi efectuată cu ajutorul unei componente a SGBD SOCRATE. Este interzisă utilizarea comenzilor de terminare forţată a prelucrărilor puse la dispoziţie de SO gazdă deoarece se poate aduce baza de date într-o stare de incoerenţă. Această interdicţie este datorată filozofiei de funcţionare a mecanismului de paginare.

În cadrul unei sesiuni de lucru un utilizator poate apela diversele componente ale SGBD SOCRATE, în general procesoare, cum ar fi :

— *procesorul* LDD (compilatorul) pentru efectuarea unor adăugiri la schema conceptuală pentru a descrie structura logică a fişierelor externe bazei de date cu care doreşte să lucreze sau a formatelor ecranelor etc. ;

— *procesorul* LMD (compilatorul) pentru testarea şi/sau execuţia programelor de cereri ;

— macrogeneratorul ;

— editorul-corector de texte pentru a descrie şi/sau întreţine textele sursă ale programelor.

În continuare vom prezenta, pentru fiecare din activităţile enunţate operaţiile specifice pe care le implică.

1) *Restaurarea componentelor software ale SGBD*

Restaurarea se efectuează de obicei în zonele destinate componentelor SO gazdă. Această operaţie este executată cu un utilitar standard al SO.

Dacă operaţia de restaurare se efectuează într-un spaţiu afectat utilizatorului atunci, în principiu, este necesară alocarea unui spaţiu fizic şi eventual iniţializarea acestuia.

Dacă caracteristicile de implementare ale SGBD SOCRATE permit, atunci operaţia de resturare poate fi urmată de o operaţie de emulare a software-ului acestuia.

la caracteristicile configurației sistemului de calcul gazdă (număr maxim de terminale, discipline de protocol utilizate, tipuri de memorii externe adresabile etc.).

Dacă software-ul rezultat nu este rezident atunci va trebui salvat pentru a constitui varianta operațională (varianta care va fi restaurată ori de cîte ori este nevoie).

2) *Generarea dicționarului datelor*

Această activitate include operații efectuate de administratorul bazei de date (ABD) cum ar fi:

— atribuirea unui nume și afectarea unui spațiu fizic;

— generarea efectivă, în acest spațiu, a dicționarului datelor. Această generare se efectuează prin intermediul unei componente speciale a SGBD.

Structura logică a acestui dicționar poate fi definită într-o schemă conceptuală parametrizată sau poate fi inclusă în logica internă a componentei.

3) *Definirea parolelor sistem de acces la dicționarul datelor*

Parolele sistem sînt destinate ABD. Cunoașterea lor permite acestuia să efectueze anumite operații asupra bazei de date cum ar fi:

— **formatare parțială sau totală a spațiului bazei de date;**
— **reorganizare parțială sau totală;**
— **declararea, ștergerea, modificarea datelor asociate utilizatorilor;**
— **colectarea de statistici privind performanțele bazeie de date etc.**

4) *Definirea caracteristicilor asociate spațiilor fizice alocate bazei de date*

Operațiile executate în cadrul acestei activități sînt:

a) alocarea spațiilor de memorie externă adresabilă;

b) descrierea caracteristicilor fiecărui spațiu alocat.

Prin această descriere se va preciza:

— numele spațiului fizic;

— numele spațiului virtual căruia i se asociază;

— caracteristicile suportului tehnic (tipul discului, numele proprietarului, volum serial etc.);

— dimensiunea spațiului fizic în unități alocabile (cilindru, bloc etc.);

— dimensiunea unei pagini;

— dimensiunea unei sub-pagini.

Dacă implementarea admite asocierea unui spațiu fizic la mai multe spații virtuale atunci acest spațiu fizic va fi divizat în spații logice.

Elementele descriptive vor fi date ca și cum spațiul logic ar reprezenta un spațiu fizic.

5) *Definirea utilizatorilor și a drepturilor de acces ale acestora*

Pentru fiecare utilizator se vor preciza elementele:

— numele, parola, drepturile de acces la operații, contul în care lucrează, spațiul (spațiile) pe care are dreptul să lucreze.

Operațiile de definire a acestor elemente sînt executate de ABD. Acesta poate interveni în orice moment, pe parcursul existenței bazei de date, pentru a modifica datele asociate utilizatorilor, a introduce noi utilizatori sau a elimina utilizatori.

6) *Formatarea spațiilor asociate bazei de date*

Operația de formatare constă în formarea unei memorii externe paginate în cadrul spațiului pe care se execută. Memoria paginată este realizată prin decuparea spațiului în pagini și subpagini cu dimensiunea specificată pentru spațiu în dicționarul datelor. Paginile și sub-paginile sînt inițializate cu valoarea zero binar și sînt înlănțuite într-o listă. Înlănțuirea se realizează cu ajutorul unui dicționar al memoriei

paginate. Acest dicționar al memoriei paginate poate fi realizat fizic utilizînd un fișier invers sau asociind fiecărui element implicat un cuvînt de control. Această listă va constitui rezervorul de spațiu din/în care se va face alocarea/recuperarea spațiului la solicitările mecanismului de paginare.

Operația de formatare poate fi efectuată pe toată baza de date, pe un spațiu al acesteia sau pe o porțiune bine determinată din acest spațiu.

În principiu această operație se efectuează o singură dată pentru o bază de date.

Ea este necesară atunci cînd se adaugă noi spații fizice la o bază de date existentă sau cînd se realizează reorganizarea unui spațiu sau a bazei de date.

7) *Compilarea schemei conceptuale a bazei de date*

Această activitate constă în lansarea în execuție a compilatorului LDD pentru analizarea schemei conceptuale descrise cu ajutorul LDD. Dacă această schemă este corectă din punct de vedere sintactic și semantic (LDD), ea va fi stocată în spațiul alocat bazei de date pentru a forma dicționarul bazei de date. Deși forma sa de reprezentare internă este dependentă de o anumită implementare, structura sa logică trebuie să permită stabilirea următoarelor elemente, pentru fiecare caracteristică:

— adresa, — identificatorul, — tipul, — lungime, condiții de validare, mod de compactare, număr maxim de apariții — filiația, — numărul intrării în dicționar, dacă este cazul etc.

Adresa sa poate fi absolută sau relativă.

Cele două forme de adresă se pot regăsi în aceeași reprezentare internă. Adresele absolute sînt alocate numai caracteristicilor definite la cel mai înalt nivel (FICHIER). Cunoașterea mecanismului de construire a acestei structuri interne, pentru o anumită implementare, permite o înțelegere aprofundată a filozofiei generale a sistemului. Prin această cunoaștere înțelegerea și aprofundarea altui mod de implementare devine mult mai ușoară.

8) *Testarea și catalogarea programelor necesare aplicațiilor*

Operațiile desfășurate în cadrul acestei activități sînt cele specifice realizării produselor informatice.

Caracteristic pentru SGBD SOCRATE este faptul că programele se cataloghează cu ajutorul componentei sale — macrogeneratorul. Catalogarea se face într-un spațiu al bazei de date.

Această activitate se desfășoară ori de cîte ori este nevoie (definirea de noi aplicații sau modificarea celor existente).

În cadrul acestei activități ABD trebuie să definească programul (programele) necesare operației de testare a coerenței logice a bazei de date. Modul de construire a acestor programe este prezentat în detaliu în capitolul 8.

9) *Încărcarea bazei de date*

Această activitate se poate realiza utilizînd:

— funcția de creare standard a SGBD SOCRATE;

— programe utilizator construite în acest scop.

Dacă implementarea nu include și funcția de creare standard atunci ABD poate utiliza tehnici de formalizare a descrierii care să-i permită construirea (utilizarea) unor produse de generare automată a programelor [ACS82, IACS83]. Aceste tehnici pot fi utilizate de altfel chiar și în cazul prezenței componentei asociate funcției de creare standard. Asupra acestor tehnici vom reveni ulterior.

10) *Întreţinerea, exploatarea şi refacerea bazei de date*

Această activitate cuprinde un ansamblu de operaţii complexe şi reprezintă activitatea desfăşurată curent asupra bazei de date.

Cîteva operaţii efectuate sînt reprezentate de :

a) actualizarea datelor (adăugare, ştergere, modificare) ;

b) interogarea bazei de date în scopul deservirii beneficiarilor aplicaţiilor funcţionale ;

c) colectarea de statistici privind funcţionarea bazei de date în scopul monitorizării performanţelor şi/sau a detectării eventualelor incoerenţe fizice ;

d) construirea „punctelor de referinţă" şi a „punctelor de control". Un punct de referinţă este definit ca o stare a bazei de date în care aceasta este coerentă din punct de vedere fizic şi logic. O bază de date este coerentă din punct de vedere fizic dacă paginile/sub-paginile alocate sînt înlănţuite corect din punct de vedere semantic şi dacă există concordanţă între informaţiile de control asociate şi starea reală a acestora. *Coerenţa logică* este specifică ansamblurilor de date definite în schema conceptuală. Practic un punct de referinţă este reprezentat de o copie a bazei de date pe suport magnetic. Această copie serveşte la refacerea acestei stări a bazei de date şi a eventualelor stări intermediare ale sale. Refacerea stărilor intermediare se realizează utilizînd fişierele jurnal construite în acest scop şi procedurile de refacere definite de ABD. Marcarea stărilor intermediare ale bazei de date se realizează prin definirea *„punctelor de control"*. Un punct de control este o stare a bazei de date considerată *coerentă* din punct de vedere fizic şi logic. Diferenţa dintre un punct de referinţă şi un punct de control este aceea că unui punct de control nu-i corespunde o copie fizică a bazei de date. Mai mult nu este necesară efectuarea întregii baterii de teste pentru determinarea coerenţei logice (această operaţie este de obicei foarte voluminoasă). Un punct de control reprezintă de fapt un punct de reluare operativă a prelucrărilor în caz de incident.

Momentul construirii unui punct de control poate diferi de la o aplicaţie funcţională la alta, de la un program la altul etc. Procedura de constituire a unui punct de control va fi definită de ABD iar construirea efectivă este lăsată şi la latitudinea utilizatorului.

11) *Reorganizarea bazei de date*

Pentru o bază de date gestionată cu SGBD SOCRATE această activitate se execută foarte rar. Efectuarea ei este necesară în următoarele două cazuri :

a) scăderea performanţelor bazei de date (creşterea timpilor de răspuns, în principal).

Această scădere a performanţelor poate fi datorată :

– saturării spaţiilor alocate (sau a unui spaţiu) peste limita de optim admisă de SO pentru un fişier ;

– apariţiei unor lanţuri lungi de sub-pagini (datorate în general volumului mare de operaţii de ştergere şi adăugare efectuate) ;

b) modificarea dimensiunii fizice a unui spaţiu şi/sau a mărimii sub-paginii.

Pentru cazul a) vom oferi în capitolul 6 un algoritm de determinare a momentului în care este necesară reorganizarea.

Indiferent care este motivul pentru care se declanşează activitatea de reorganizare, aceasta trebuie să cuprindă următoarele operaţii (pentru un spaţiu) :

1°. salvarea datelor stocate în acest spaţiu cu ajutorul utilitarului de reorganizare ;

2°. definirea noilor caracteristici ale spaţiului (modificarea definirii corespondente din dicţionarul datelor);

3°. formatarea spaţiului conform specificaţiilor din dicţionarul datelor;

4°. restaurarea cu ajutorul utilitarului de reorganizare a datelor salvate la pasul 1°.

Pentru aceste operaţii este impusă respectarea acestei ordini de efectuare a lor Fiecare operaţie particulară poate fi efectuată simultan pentru mai multe spaţii.

Implementări ale dicţionarului datelor

Pentru SOCRATE V1.5 şi V1.6.R dicţionarul datelor este gestionat cu ajutorul „bazei de baze".

Baza de baze este definită cu LDD ca o schemă conceptuală SOCRATE. Spaţiul real alocat are aceeaşi structură logică cu cel al unei baze de date. Pentru generarea acestei baze de baze ABD utilizează utilitarul xxxGBDB (unde xxx este SOC pentru V 1.5 şi LINK pentru V 1.6.R). Acest utilitar realizează:

— formatarea spaţiilor bazei de date conform specificaţiilor furnizate de ABD;

— compilarea schemei conceptuale şi obţinerea dicţionarului bazei de baze.

Numele de bază de baze vine de la faptul că aceasta este capabilă să gestioneze dicţionarul datelor pentru un număr foarte mare de baze de date (maxim $2^{24} - 1$).

Efectuarea gestiunii simultane a mai multor dicţionare de date este necesară pentru a permite exploatarea simultană a mai multor baze de date. Această utilizare simultană este permisă în cazul exploatării cu ajutorul metodei de acces (maxim 255 baze active) sau în cazul exploatării multibază în regim conversaţional (maxim 20 baze active.

Spaţiul bazei de baze conţine şi zone de manevră destinate procesoarelor.

O bază de date gestionată cu ajutorul acestor versiuni are un spaţiu fizic unic (un fişier de organizare nedefinită în accepţiunea noţiunii utilizate pentru SO SIRIS/HELIOS). Acest spaţiu fizic este decupat în spaţii logice.

Numele şi destinaţia acestor spaţii logice sînt:

a) pentru V 1.5:

— FICH destinat stocării:
  - dicţionarului bazei de date;
  - datelor atribuite caracteristicilor care compun structura;
  - codului sursă (în format de reprezentare internă) al macro şi programelor precompilate;

— DICO este o resursă partajată pentru diversele dicţionare ale datelor (operator de transformare) astfel:
  - dicţionarul asociat numelor de programe (macro, precompilate, IMT);
  - dicţionarele asociate caracteristicilor declarate chei de acces (tabela de indecşi). Pentru fiecare declaraţie de cheie se rezervă un dicţionar. Numărul de intrări al dicţionarului alocat se determină în funcţie de tipul şi dimensiunea caracteristicii;

— PROG destinat stocării codului executabil al programelor precompilate şi IMT;

b) pentru V 1.6.R

– STRU destinat stocării dicționarului bazei de date ;

– FICH destinat stocării datelor atribuite caracteristicilor definite în structură;

– DICO pentru stocarea dicționarelor asociate :

- numelor de programe ;
- caracteristicilor declarate chei de acces ;
- legăturilor de tip inel-referire (ANNEAU-REFERE) ;

– PROG pentru stocarea :

- codului sursă al macro și programelor precompilate ;
- codului executabil al programelor precompilate și IMT ;

– TRAV destinat ca zonă de manevră pentru procesoare ;

– MSRT destinat ca zonă de manevră în cazul utilizării sortării.

Reguli de prezentare a limbajelor

Pentru descrierea sintaxei limbajelor prezentate se va utiliza un limbaj formal (BNF – Backus Naur Form), *meta-limbaj*, care utilizează simboluri speciale (*meta-simboluri*). Gramatica acestui limbaj formal, definită ca ansamblul regulilor specifice (*producții*) destinate descrierii, face apel la un set specific de reprezentări (*alfabetul limbajului*) și la reguli care manipulează două tipuri de simboluri :

– *terminale* : șiruri de caractere ale alfabetului limbajului care se utilizează conform definirii lor în toate regulile de producere în care apar ;

– *neterminale* : simboluri definite prin altă regulă de producere. Aceste simboluri sînt noțiuni pe care trebuie să le înlocuim, acolo unde apar, prin efectuarea producerilor asociate tuturor simbolurilor neterminale, pînă la obținerea unui simbol terminal.

Meta-simbolurile utilizate sînt de două tipuri :

– de bază ;

– adiționale.

1. *Meta-simbolurile de bază*, proprii BNF, permit descrierea integrală a tuturor regulilor de producție și sînt următoarele :

⟨ ⟩ – delimitează un simbol neterminal ;

: : = – caracteristica regulei de producere cu semnificația „produs de" sau „desemnat de". Simbolul se plasează între noțiunea de descris (stînga) și primul mod de producere (dreapta) ;

/ – separă două moduri de producere posibile („sau") ;

Exemplu :

⟨simbol⟩ : : = ⟨terminal⟩/⟨neterminal⟩

Această regulă de producere se interpretează astfel : „un *⟨simbol⟩* este desemnat de un *⟨simbol⟩* *⟨terminal⟩* sau un *⟨simbol⟩* *⟨neterminal⟩*".

2. *Meta-simbolurile adiţionale* sînt utilizate în scopul reducerii volumului descrierii producerilor şi sînt următoarele :

[ ] — arată prezenţa opţională a unui simbol care apare într-o regulă de producere ;

⟨*simbol*$_1$⟩
⟨*simbol*$_2$⟩
.
.
.
⟨*simbol*$_n$⟩ } : : = ⟨*regula de producere*⟩ — simbolurile prezentate în stînga acoladei au aceiaşi regulă de producere ;

{*var*$_1$/*var*$_2$/.../*var*$_n$} — o regulă de producere conţine un simbol terminal cu mai multe variante de realizare ;

sau

var$_1$
var$_2$
.
.
.
var$_n$ }

max. r — în cazul regulilor de definire recursive care prezintă o limită de recursivitate se va indica această limită, prin *max. r*, în partea dreaptă a definiţiei ;

— pentru a familiariza cititorul cu simbolurile terminale care reprezintă cuvinte cheie ale limbajului acestea vor fi scrise cu majuscule şi subliniate. La această regulă vom renunţa treptat prezentînd exemplele în forma lor naturală de producere.

În conformitate cu regulile enunţate anterior se va defini alfabetul utilizat de procesoarele şi limbajele descrise :

⟨alfabet⟩ : : = ⟨literă⟩/⟨cifră⟩/⟨caracter special⟩
⟨literă⟩ : : = A/B/C/D/E/F/G/H/I/J/K/L/M/N/O/P/Q/R/S/T/U/V/W/X/Y/Z
⟨cifră⟩ : : = 0/1/2/3/4/5/6/7/8/9
⟨caracter special⟩ : : = ./,/:/=/+/-/*/;/$/</>/¬/#/(/)/!/&/∂/%/'/ƀ,?/
ƀ : : = spaţiu

În cadrul prezentării vom mai utiliza notaţiile :

— R — mulţimea numerelor reale ;
— N — mulţimea numerelor naturale ;
— I — mulţimea numerelor întregi ;
— U — reuniunea a două mulţimi ;
— ∩ — intersecţia a două mulţimi ;
— ∈ — apartenenţa la un interval ;
 • (V_1, V_2) — deschis (exclusiv V_1 şi V_2) ;
 • [V_1, V_2) — închis la stînga (inclusiv V_1) ;
 • (V_1, V_2] — închis la dreapta (inclusiv V_2) ;
 • [V_1, V_2] — închis (inclusiv V_1 şi V_2) ;
— X'cc' — valoare în hexazecimal.

Pentru a familiariza cititorul cu limbajele LDD şi LMD vom prezenta în continuare o listă a principalelor cuvinte cheie utilizate. Această listă formează, de altfel, un dicţionar destinat celor care nu cunosc limba franceză în care sînt exprimate aceste cuvinte cheie. Pentru cuvintele cheie care sînt date ca abrevieri vom prezenta în paranteză cuvîntul (cuvintele) care a stat la baza formării lor.

A — la
ALORS — atunci
ANNEAU — inel
ARRIERE — înapoi
AVANT — înainte
AVEC — cu
AYANT — avînd
BINAIRE — binar
BLOQUER — blochează
C — creează (creation)
CHAINE — lanţ
CLASSE — clasează (ordonează)
CLE — cheie
D — început/număr/defineşte(debut,denomber,definition)
DANS — în
DE — de la
DEPUIS — după
DEBUT — început
DECIMAL — zecimal
DILATE — despachetat
DONT — a cărui
DOUBLE — dublă
ECRIRE — scrie
EN — în
ENCOURS — în curs, în lucru
ENTITE — entitate
ERREUR — eroare
ET — şi
EXEC — execută
EXISTE — există
EXT — extern/exterior (externe)
FAIRE — execută
FIN — sfîrşit
FORMAL — formal
FORMAT — format
G — generează (generer)
I — imprimă (impression)
INVERSE — invers
LIBERER — eliberează
LIRE — citeşte
M — modifică (mise a jour)
MOT — cuvînt
NON — nu/negat
NUMDE — numărul lui (*NUM*ero DE)
ORDONNE — ordonat
OU — sau
PACKE — împachetat
PAR — prin ordinea valorilor
PAUSE — pauză
PAS — nu există
POUR — pentru
REFAIRE — reia
REFERE — referire
S — şterge(suppresione)
SE — şterge ansamblu (suppresion de l'ensamble)
SI — dacă
SIMPLE — simplu
SINON — în caz contrar/altfel
SORTIE — ieşi din
SUIVANT — următorul
TELQUE — oricare
TEXTE — text
TOUT,TOUTE — toţi, toate
U — nedefinit (undefined)
UN,UNE — unul, una
UNIQUE — unica.

Concluzii

Capitolul doi face o prezentare generală a noţiunilor SGDB-SOCRATE, noţiuni ce vor fi detaliate în capitolele următoare.

Cititorul face cunoştinţă cu : istoricul acestui produs, cu versiunile lui, cu componentele lui şi noţiunile caracteristice fiecăreia din ele precum şi cu activităţile pe care trebuie să le realizeze utilizatorii de-a lungul ciclului de viaţă al unei baze de date SOCRATE. Atragem atenţia cititorului că structura ciclică prezentată în acest capitol este o structură deosebit de puternică. Acest tip de structură nu este acceptat de SGBD-urile de tip CODASYL. Pentru utilizarea ei oferim mai multe informaţii în capitolul 4. Remarcăm faptul că această structură este structura de bază utilizată la SOGER [S1].

Din acest moment cititorul poate să treacă la studiul în detaliu al uneia sau alteia dintre componentele SGBD-SOCRATE. Capitolele următoare prezintă, fiecare, în detaliu cîte o componentă a SGBD-SOCRATE sau cîte o problematică a bazei de date SOCRATE.

Pentru a studia componentele cititorul trebuie să se familiarizeze cu notatiile şi regulile de prezentare din acest capitol. În scopul familiarizării sale cu limbajele de descriere şi manipulare am considerat oportună prezentarea, în acest moment, a principalelor cuvinte cheie utilizate de acestea.

SECŢIUNEA II

SGBD SOCRATE — Tratare generală

3. LIMBAJUL DE DESCRIERE A DATELOR. LDD-SOCRATE

Introducere

Limbajul de descriere a datelor permite:

— definirea informaţiilor elementare, care caracterizează obiectul sau fenomenul supus analizei, prin stabilirea identificatorului şi a structurii lor;

— precizarea tipului de acces dorit;

— definirea relaţiilor de agregare existente între aceste date elementare conform unei structuri arborescente sau în reţea.

Formatul de descriere a datelor apare astfel:

⟨*informaţie-elementară*⟩ : : = ⟨*identificator*⟩⟨*tip*⟩

⟨*identificator*⟩ : numele sub care este recunoscută informaţia respectivă. Se formează prin combinarea a cel mult 30 caractere alfanumerice (litere şi cifre), combinare la libera alegere a utilizatorului, şi caracterul special '—' (liniuţă) care nu trebuie să fie primul sau ultimul caracter al identificatorului. Identificatorul poartă denumirea de *nume de caracteristică* (nume de dată).

⟨*tip*⟩ : este elementul care precizează informaţia desemnată de identificator;

⟨*tip*⟩ : : = ⟨*d*⟩⟨*v*⟩⟨*m*⟩⟨*s*⟩⟨*c*⟩⟨*a*⟩⟨*o*⟩

⟨*d*⟩ : genul de valori pe care le pot lua datele

⟨*v*⟩ : condiţiile de validare pe care trebuie să le îndeplinească un anumit gen de valoare pentru a fi acceptat de sistem;

⟨*m*⟩ : spaţiul de memorie ocupat de informaţia elementară şi modul de calcul al acestuia;

⟨*s*⟩ : structura şi semnificaţia spaţiului de memorie ocupat;

⟨*c*⟩ : cadrajul cerut de tipul de informaţie ales;

⟨*a*⟩ : modul de acces permis să se efectueze asupra datelor respective;

⟨*o*⟩ : genul de operaţii autorizat să se efectueze cu datele respective.

În continuare se vor preciza cîteva specificaţii în legătură cu fiecare parametru, astfel:

⟨*d*⟩ : : = ⟨*adrese*⟩/⟨*valori numerice*⟩/⟨*şir caractere alfanumerice*⟩/⟨*bit*⟩

⟨*v*⟩ : : = ⟨*apartenenţă la un interval de valori*⟩/⟨*coincidenţă de tip*⟩/⟨*apartenenţă la un dicţionar*⟩

⟨*m*⟩ : : = ⟨*multiplu de cuvînt*⟩/⟨*cuvînt*⟩/⟨*octet*⟩/⟨*bit*⟩

⟨*s*⟩ : : = ⟨*informaţii de sistem*⟩/⟨*informaţii utilizator*⟩

⟨c⟩ : : = ⟨*adresă sub pagină*⟩/⟨*adresă cuvînt*⟩/⟨*adresă octet*⟩/⟨*adresă bit*⟩
⟨a⟩ : : = ⟨*secvențial*⟩/⟨*direct*⟩/⟨*secvențial indexat*⟩
⟨*secvențial*⟩ : : = ⟨*parcurgere naturală*⟩/⟨*parcurgere inversă*⟩/⟨*parcurgere completă*⟩
⟨*direct*⟩ : : = ⟨*prin număr de ordine*⟩/⟨*prin dicționar*⟩
⟨o⟩ : : : = ⟨*comparații**⟩/⟨*calcule aritmetice*⟩/⟨*funcțiunii speciale*⟩/⟨*desemnare de relație filială*⟩/⟨*creare*⟩/⟨*ștergere*⟩/⟨*adăugare*⟩/⟨*modificare*⟩/⟨*acces la date*⟩.

În descrierea structurii bazei de date, LDD face referire la o serie de noțiuni, cărora li se poate găsi un „echivalent" în limbajele clasice, cum ar fi :

— *caracteristică* : are semnificația de tip de dată (un tip de dată complex) ;

— *entitate* : poate fi asociată cu noțiunea de descriere a structurii logice a unui fișier din organizarea clasică (acest fișier poate fi secvențial, indexat secvențial, direct multiindexat ; poate fi parcurs într-o multitudine de ordini (sau combinații de moduri, de parcurgere) iar relațiile structurale incluse între elemente pot fi arborescente sau în rețea) ;

— *realizare* : similar unei înregistrări dintr-un „fișier" (un caz concret de valori asociat unei descrieri complete de entitate).

Pentru sistemul de calcul, o bază de date descrisă cu ajutorul LDD-SOCRATE reprezintă un fișier, în accepțiunea clasică a noțiunii, de organizare nedefinită (deci sînt admise toate funcțiunile de manipulare a datelor puse la dispoziție de sistem). Odată definită, structura unei baze de date, reprezintă modelul de referință utilizat la orice consultare a datelor pe care le definește și le conține.

Sintaxa de definire a unei structuri, descrisă cu LDD SOCRATE, poate fi exprimată în BNF astfel :

⟨*fișier*⟩ : : = ⟨*bloc*⟩
⟨*bloc*⟩ : : = ⟨*identificator*⟩ DEBUT ⟨*lista de caracteristici*⟩ FIN. Pentru blocul „fișier" identificatorul este atribuit în mod automat de către sistem și are valoarea FICHIER.
⟨*lista de caracteristici*⟩ : : = ⟨*caracteristică*⟩/⟨*caracteristică*⟩⟨*lista de caracteristici*⟩
⟨*caracteritiscă*⟩ : : = ⟨*caracteristică simplă*⟩/⟨*caracteristică de relație structurală*⟩
⟨*caracteristică simplă*⟩ : : = ⟨*cuvînt*⟩/⟨*lista-de-valori*⟩/⟨*valoare numerică*⟩/⟨*text*⟩/⟨*zecimal*⟩
⟨*cuvînt*⟩ : : = ⟨*identificator*⟩ MOT [[(]⟨*n*⟩[)]] [⟨*declarație* $acces_1$⟩]
⟨*n*⟩ ∈ [1,30] ∩ N
⟨*declarație* $acces_1$⟩ : : = AVEC CLE [{UNIQUE/DOUBLE AVANT/DOUBLE ARRIERE}]
[{ORDONNE/NON ORDONNE}] [([⟨*index*⟩] [$ *prog*⟩])] FIN
⟨*lista-de-valori*⟩ : : = ⟨*identificator*⟩ (⟨n_1⟩⟨*sep*⟩[⟨n_2⟩])
(⟨*valoare*$_1$⟩[⟨*sep*⟩⟨*valoare*$_2$⟩⟨*sep*⟩...⟨*valoare*$_i$⟩...])
[⟨*declarație* $acces_2$⟩]
⟨n_1⟩ ∈ [1,255] ∩ N, ⟨n_2⟩ ∈ [1,30] ∩ N
⟨*declarație* $acces_2$⟩ : : = AVEC CLE [{UNIQUE/DOUBLE AVANT/DOUBLE ARRIERE}] [{CHAINE SIMPLE/CHAINE DOUBLE}]FIN
⟨*valoare numerică*⟩ : : = ⟨*identificator*⟩DE[⟨*semn*⟩]⟨n_1⟩A[⟨*semn*⟩]⟨n_2⟩
[[(]⟨n_3⟩[)]][⟨*declarație* $acces_1$⟩]
⟨n_1⟩, ⟨n_2⟩, ⟨n_3⟩ ∈ $[-(2^{31}-1), +(2^{31}-1)] \cap I$ și $n_1 \leqslant n_2 \leqslant n_3$

* Operația denumită generic „comparație" este utilizată și pentru a desemna operația de atribuire.

⟨*zecimal*⟩ : : = ⟨*identificator*⟩ DECIMAL [(]⟨n_1⟩[V ⟨n_2⟩][)]][*declarație acces*$_1$]
unde : $n_1 \in [0,15] \cap N$; $\langle n_2 \rangle \in [0,7] \cap \overline{N}$
text : : = *identificator* TEXTE [(] n_1 [n_2] [)]
$n_1 \in [1,255] \cap N$, $n_2 \in [1,60] \cap N$
⟨*caracteristică de relație structurală*⟩ : : = ⟨*implicit*⟩/⟨*explicită*⟩
⟨*implicită*⟩ : : = ⟨*bloc*⟩/⟨*entitate*⟩
⟨*entitate*⟩ : : = ENTITE [(]⟨*n*⟩[)]⟨*bloc*⟩
$\langle n \rangle \in [1,(2^{24} - 1)] \cap N$
⟨*explicită*⟩ : : = ⟨*inel*⟩/⟨*referire*⟩/⟨*invers*⟩
⟨*inel*⟩ : : = ⟨*identificator*⟩ ANNEAU [AVEC ⟨*tip-legătură*⟩ FIN]
⟨*tip-legătură*⟩ : : = CHAINE SIMPLE/CHAINE DOUBLE
⟨*referire*⟩ : : = ⟨*referire simplă*⟩/⟨*referire cu inel*⟩
⟨*referire simplă*⟩ : : = ⟨*identificator*⟩ REFERE {UN/UNE}⟨*nume entitate*⟩
[⟨*calificarea*⟩]
⟨*referire cu inel*⟩ : : = ⟨*identificator*⟩ REFERE ⟨*nume inel*⟩ DE
{UN/UNE}⟨*nume entitate*⟩ [⟨*calificare*⟩]
[AVEC ⟨*tip-legătură*⟩ FIN]
⟨*calificare*⟩ : : = DE {UN/UNE} ⟨*nume entitate*$_1$⟩ [⟨*calificare*⟩]/ DE
⟨*nume bloc*$_j$⟩[⟨*calificare*$_e$⟩] DE {UN/UNE} ⟨*nume entitate*$_f$⟩
⟨*invers*⟩ : : = ⟨*identificator*⟩ INVERSE {TOUT/TOUTE}
⟨*nume entitate*⟩[⟨*calificare*$_i$⟩]
⟨*calificare*$_i$⟩ : : = DE {TOUT/TOUTE} ⟨*nume entitate*$_i$⟩ [⟨*calificare*$_i$⟩]
DE {UN/UNE} ⟨*nume entitate*$_f$⟩/
DE ⟨*nume bloc*$_j$⟩ [⟨*calificare*$_i$⟩] DE UN/UNE}
⟨*nume entitate*$_f$⟩
⟨*separatorii de limbaj*⟩ : : = (|)|=|;|'|<|>|ƀ
ƀ : = *spațiu*.

În afara acestor elemente, care fac parte integrantă din structura internă a bazei de date, sistemul mai admite utilizarea unor elemente „*fictive*" (care sînt externe structurii), elemente prin care este posibilă descrierea unor fișiere externe bazei de date (fișiere considerate în accepțiunea clasică a cuvîntului, de tip COBOL, de exemplu, prin care se pot efectua operații asupra structurii interne (încărcarea bazei de date pornind de la fișiere de tip COBOL, extragerea de date, descrierea structurii ecranelor dispozitivelor de afișare etc.).

⟨*caracteristici fictive*⟩ : : = ⟨*formal logic*⟩
⟨*formal logic*⟩ : : = FORMAL ⟨*bloc formal*⟩
⟨*bloc formal*⟩ : : = ⟨*identificator*⟩ DEBUT ⟨*lista caracteristicii formal*⟩ FIN
⟨*lista caracteristicii formal*⟩ : : = ⟨*cuvînt formal*⟩/⟨*zecimal împachetat*⟩
⟨*zecimal despachetat*⟩/⟨*formal logic repetitiv*⟩/
⟨*zecimal împachetat cu virgulă*⟩/⟨*zecimal despachetat cu virgulă*⟩/⟨*binar*⟩
⟨*cuvînt formal*⟩ : : = ⟨*identificator*⟩ MOT [(] [⟨n_1⟩] ⟨n_2⟩ [)]
⟨*zecimal despachetat*⟩ : : = ⟨*identificator*⟩ DILATE [(] ⟨n_1⟩ ⟨n_2⟩ [)]
⟨*zecimal împachetat*⟩ : : = ⟨*identificator*⟩ PACKE [(] ⟨n_3⟩ [)]
⟨*formal logic repetitiv*⟩ : : = FORMAL [(] ⟨*n*⟩ [)] ⟨*bloc formal*⟩
$\langle n \rangle \in [1,(2^{24} - 1)] \cap N$, $\langle n_2 \rangle \in [1,80] \cap N$, $n_1 \in [1, \langle l\ form \rangle] \cap N$
⟨*zecimal despachetat cu virgulă*⟩ : : = ⟨*identificator*⟩ DILATE
[(] [⟨n_1⟩] ⟨n_2⟩ [V ⟨n_3⟩] [)]
⟨*zecimal împachetat cu virgulă*⟩ : : = ⟨*identificator*⟩ PACKE
[(]][⟨n_1⟩] ⟨n_2⟩ [V ⟨n_3⟩] [)]

unde : ⟨n_1⟩ ∈ [1, ⟨*l form*⟩], ⟨n_2⟩ ∈ [0,15] ∩ N, ⟨n_3⟩ ∈ [0,7] ∩ N ;
⟨*binar*⟩ : : = ⟨*identificator*⟩ BINAIRE [(] [⟨n_1⟩] ⟨n_2⟩ [)]
⟨n_2⟩ : : = 2/4, ⟨n_1⟩ ∈ [1, ⟨*l form*⟩] ∩ N

În paragrafele următoare vor fi oferite specificaţii privind utilizarea diverselor elemente prezentate în descrierea obiectelor manipulate de LMD şi vor fi detaliate şi prezentate regulile de producţie ale tuturor simbolurilor neterminale care nu au fost prezentate în această sinteză.

La prezentare facem precizarea că pentru caracteristicile la care apare un * la *cadraj* aceasta înseamnă că adresa de cadraj va fi forţată automat la multiplu de cuvînt dacă caracteristica apare imediat după o declaraţie de ⟨*bloc*⟩ sau ⟨*entitate*⟩.

Observaţiile prezentate în paragrafele următoare sînt destinate, în principal, celor care au depăşit stadiul de iniţiere în SGBD-SOCRATE.

Caracteristici

Caracteristica de tip ⟨cuvînt⟩ (MOT)

Rol : descrierea atributelor (cîmpurilor) alfanumerice ;

Sintaxa :

⟨*identificator*⟩ MOT [(⟨n⟩)/⟨n⟩] [⟨*declaraţie acces*$_1$⟩]
unde ⟨*n*⟩ ∈ [1,30] ∩ N

⟨*n*⟩ : specifică lungimea (în octeţi) pe care dorim să o rezervăm pentru informaţia desemnată de identificator. Deoarece declararea lui ⟨*n*⟩ este opţională, valoarea sa implicită este de 30 octeţi ;

MOT : defineşte un şir de caractere alfanumerice oarecare ;

Date acceptate : orice şir de caractere alfanumerice cu lungimea ⩽ *n* ;

Condiţii de validare : nu există restricţii asupra caracterelor introduse. Deoarece anumite caractere au o semnificaţie specială pentru unele funcţiuni puse la dispoziţia utilizatorului de către SGBD-SOCRATE, este indicată utilizarea lor cu prudenţă. Aceste caractere sînt :

— ' (apostrof) : din cauza utilizării valorilor între două ' ;

— / (slash) : datorită semnificaţiei pe care o are în cazul utilizării funcţiei de „creare standard" (fişier slash) sau a funcţiunilor „bibliotecarului editorului de texte" ;

— U şi NON : dacă caracteristica este pusă la zi în conversaţional ;

— spaţiul : este separator de limbaj de declararea modelelor de apel ale macroinstrucţiunilor.

Dacă şirul de caractere introdus este > n atunci va fi trunchiat la dreapta (se iau în considerare doar primele ⟨n⟩ caractere din stînga) ;

Spaţiul ocupat : (n + 1) octeţi.

Structura şi semnificaţia spaţiului ocupat :

— octetul 1 : conţine numărul k ⩽ n de caractere introduse (inclusiv spaţii) ;

— următorii k octeţi : conţin şirul de caractere introdus codificat în formatul intern de reprezentare (EBCDIC sau ASCII) ;

— restul de (n − k) octeţi (n − k ≠ 0) : conţin caracterul binar X'00' (nedefinit) ;

Exemple :

NUME MOT
NUME MOT 30
NUME MOT (30)

Aceste descrieri sînt identice ca semnificaţie. Necesarul de memorie pentru caracteristica NUME este de 30+1 = 31 octeţi.

Structura zonei de memorie rezervate este :

| 1 | 2 | 3 | . . . . |31 |

↑ contor — caractere care urmează să fie reprezentate

Valoarea caracterului ⟨contor⟩ indică numărul efectiv de caractere introduse (k).

De exemplu, după efectuarea atribuirii :

NUME : = 'ACADEMIA DE STUDII ECONOMICE', **structura caracteristicii NUME va fi, de exemplu, codificat în EBCDIC.**

1C	C1	C3	C1	C4	C5	D4	C9	C1	40	C4	C5	40	E2	E3	E4	C5	C9	C9
↑	A	C	A	D	E	M	I	A		D	E		S	T	U	D	I	I

Lcontor = 28 în binar

40	C5	C3	D6	D5	D6	D4	C9	C3	C5	C0	00
	E	C	O	N	O	M	I	C	E		

Observaţie : spaţiul de memorie rezervat în memoria virtuală poate să aibă drept corespondent în spaţiul real o structură gestionată de utilizator.

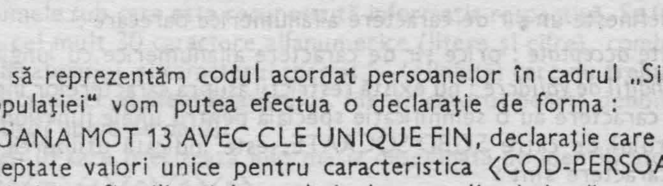
Exemplu :

Dacă dorim să reprezentăm codul acordat persoanelor în cadrul „Sistemului de evidenţă a populaţiei" vom putea efectua o declaraţie de forma :

COD-PERSOANA MOT 13 AVEC CLE UNIQUE FIN, declaraţie care specifică faptul că sînt acceptate valori unice pentru caracteristica ⟨COD-PERSOANA⟩ şi că această caracteristică va fi utilizată drept cheie de acces discriminată.

Structura informaţiei reprezentate în această caracteristică este :

COD-PERSOANA ⟨S, AA, LL, ZZ, NNNNN, C⟩



iar valorile atribuite vor fi numere în format zecimal despachetat.

Gestiunea elementelor componente revine în sarcina utilizatorului şi va fi realizată cu ajutorul instrucţiunilor LMD.

Cadraj : octet*.

Operaţii admise : comparaţii între şiruri de caractere alfanumerice oarecare şi teste de existenţă.

Caracteristica de tip ⟨listă-de-valori⟩

Rol : descrierea atributelor (caracteristicilor) alfanumerice cu valori predefinite;

Sintaxa :

⟨*identificator*⟩(⟨n_1⟩ ⟨*sep*⟩[⟨n_2⟩])(⟨*valoare*$_1$⟩⟨sep⟩ ⟨*valoare*$_2$⟩⟨*sep*⟩...⟨*valoare*$_i$⟩...
...) [⟨*declarație acces*$_2$⟩]

unde :

$\langle n_1\rangle \in [1,2^{16}-1] \cap N$; $\langle n_2\rangle \in [1,30] \cap N$;

⟨n_1⟩ : definește numărul maxim posibil de elemente ⟨*valoare*$_i$⟩, prevăzut pentru listă ($i \in [1, n_1]$) ;

⟨n_2⟩ : definește numărul maxim de caractere prevăzut pentru un element ⟨*valoare*$_i$⟩ din listă. În cazul în care nu este declarat, ia implicit valoarea 30 ;

⟨*valoare*$_i$⟩ : șir de caractere, exclusiv separatorii de limbaj, ales de utilizator pentru caracterizarea obiectului real ;

⟨*sep*⟩ : reprezintă unul din separatorii de limbaj desemnați pentru a delimita elementele ⟨*valoare*$_i$⟩ din listă.

Spațiul este considerat ca separator implicit.

Observații :

În funcție de relația de ordine care există între numărul maxim de valori posibile n_1 și numărul efectiv n_e de elemente valoare$_i$ declarate, listele sînt :

— *deschise* : dacă $n_1 > n_e$. În acest caz este admisă redefinirea formală a acestui element din structură (prin adăugarea unui număr de elemente ⟨*valoare*$_i$⟩ mai mic sau egal cu $(n_1 - n_e)$), fără afectarea organizării datelor pe suportul real ;

— *închise* : dacă $n_1 = n_e$. În acest caz, orice redefinire a acestui element din structură necesită reorganizarea datelor de pe suportul real.

Date acceptate : un șir de caractere alfanumerice care coincide cu unul din elementele ⟨*valoare*$_i$⟩ declarate ; din acest punct de vedere, caracteristica de tip ⟨*listă-de-valori*⟩ apare ca o restricție a caracteristicii de tip MCT ;

Condiții de validare : sistemul verifică și acceptă numai acele date care aparțin listei de elemente ⟨*valoare*$_i$⟩ declarate.

În cazul în care se încearcă introducerea unei date nedeclarate ca element al listei, sistemul nu modifică informația și semnalează, pe suportul de ieșire, incidentu produs. În funcție de modul de lucru ales de utilizator, pentru prelucrare, la semnalarea unui incident de acest gen se încearcă corectarea valorii, prin cererea ei la dispozitivul de intrare (conversațional), sau se ignoră (batch processing) și prelucrarea continuă.

Spațiul ocupat : numărul de biți necesari (NBnec) pentru a codifica în binar numărul de valori declarate n_1.

(1) NBnec reprezintă cea mai mică valoare întreagă pentru care $_2$NBnec $\geq n_1$; NBnec $\in [1,16]$.

Structura și semnificația spațiului ocupat : numărul de ordine, în binar, al valorii din listă ;

Cadraj : *bit**.

Operații admise : comparații între șiruri de caractere alfanumerice oarecare, teste de existență, operand al funcției D (determinarea numărului i al valorii atribuite listei de valori) a LMD.

În zona asociată listei de valori nu se vor reprezenta elementele din listă ci numai pozițiile lor în listă. Elementele se memorează o singură dată iar la interogare cu ajutorul poziției în listă se realizează decodificarea și acestea apar în clar.

Observații :

1. Dacă $2^{NBnec} > n_1$ (NBnec luat conform condiției (1)), atunci este permisă modificarea formală a structurii prin inlocuirea lui n_1 cu cel mai mare număr n_{max} care indeplinește condiția $n_{max} \leqslant 2^{NBnec}$ in acest caz, spațiul de memorie afectat caracteristicii rămine nemodificat iar lista de valori devine deschisă.
2. Orice redefinire a unei liste de valori trebuie să păstreze ordinea de definire a valorilor anterioare.
   Dacă se dorește, utilizatorul este obligat să salveze aceste valori pe un fișier (poate fi de tip COBOL sau se pot utiliza eventualele caracteristici de cadraj din structură). După această salvare se efectuează modificarea dorită (respectind numai condiția de spațiu ocupat) și se recreează valorile reale ale caracteristicii pornind de la fișierul obținut anterior (prin punere la zi).

Exemple :

1. GRUPA-SANGUINA (4 4) OI AII BII ABIV)
   ↓ (al doilea 4) lungimea maximă a unui element valoare ;
   ↓ (primul 4) Nbnec ; 3 biți (atenție : cu 2 biți pot reprezenta 4 valori dar cu numelele 0, 1, 2, 3, deci Nbnec este 3 pentru a putea reprezenta valoarea 4).

Observație :

Deoarece $2^{Nbnec} > 4$ ($2^3 = 8$) rezultă că această listă este deschisă și este admisă adăugarea de noi valori. În memoria virtuală elementele listei de valori sint reprezentate ca perechi ⟨*lungime*$_1$⟩, ⟨*valoare*$_1$⟩, unde ⟨*lungimea*$_1$⟩ $\in [1, \langle n_2 \rangle]$, și specifică numărul real de caractere al valorii.

Pentru exemplul precedent, reprezentarea în memoria virtuală (BROU) este :

B	O I I I	C	A I I I I	C	B I I I I	D	A	B	I	V

2. SEX (2;1) (B;F)
3. APARTENENTA-POLITICA (15 4) (PCR UTC ODUS FARA)
4. NATIONALITATEA
   (32 1) (1 2 3 4 5 6 7 8 9
   A B C D E F G H I X)

Caracteristica de tip ⟨valoare numerică⟩

Rol : descrierea atributelor (cîmpurilor) numerice întregi ;

Sintaxa :

⟨*identificator*⟩ DE [⟨*semn*⟩] ⟨n_1⟩ A [⟨*semn*⟩] ⟨n_2⟩
[(⟨n_3⟩)/⟨n_3⟩] [⟨*declarație acces*$_1$⟩]

unde :

⟨n_1⟩, ⟨n_2⟩, ⟨n_3⟩ $\in [-(2^{31}-1), +(2^{31}-1)] \cap I$ și satisfac relația de ordine $n_1 \leqslant n_2 \leqslant n_3$;

semn : : = +/– ; semnul + este implicit ;

– DE — valoarea minimă (De la) ;

– A — valoarea maximă (la) ;

⟨n_1⟩ : marginea inferioară a intervalului de variație a valorii ;

⟨n_2⟩ : marginea superioară a intervalului de variație a valorii ;

⟨n_3⟩ : valoarea maximă, potențială, prevăzută pentru marginea superioară a intervalului de variație.

Date acceptate :

— valori numerice întregi care pot lua valori în intervalul $[\langle n_1 \rangle, \langle n_3 \rangle]$, declarat. În funcţie de prezenţa sau absenţa limitei $\langle n_3 \rangle$, intervalul de variaţie este, respectiv :

— *deschis* : permiţînd o redefinire formală a caracteristicii, fără afectarea organizării datelor pe suportul real, dînd lui $\langle n_2 \rangle$ orice valoare din cele cuprinse în intervalul $[n_2, n_3]$;

— *închis* : redefinirea formală implică reorganizarea datelor pe suportul real.

Condiţii de validare :

— se autorizează numai acele valori care aparţin intervalului $[n_1, n_2]$. Incidentele apărute la prelucrare sînt semnalate pe suportul de ieşire permiţînd reluarea dialogului (conversaţional), pentru corectarea valorii, sau ignorarea (batch processing).

Spaţiul ocupat : numărul de biţi necesari (NBnec) pentru a codifica în binar „plaja de variaţie" (pv) declarată pentru valoare.

$$pv ; \begin{cases} |n_2 - n_1|, \text{ dacă } n_3 \text{ nu este declarat ;} \\ |n_3 - n_1|, \text{ dacă } n_3 \text{ este declarat ;} \end{cases}$$

NBnec este cea mai mică valoare întreagă pentru care $2^{NBnec} \geq pv$.

Structura si semnificaţia spaţiului ocupat : deplasarea, în binar, a valorii introduse faţă de marginea inferioară n_1, a intervalului de variaţie (este întotdeauna o valoare pozitivă).

Cadraj : depinde de relaţia de ordine care există între numărul de biţi necesari (NBnec), pentru codificarea în binar a plajei de variaţie, şi de numărul de biţi care rămîn liberi (NB_1) într-un cuvînt de memorie în care s-ar introduce această valoare.

Astfel avem :

— $NB_1 = 32 - NBnec$ (32 = numărul de biţi dintr-un cuvînt de memorie)

$$\text{— dacă : } \begin{cases} NBnec \geqslant NB_1, \text{ cadraj la nivel de cuvînt ;} \\ NBnec < NB_1, \text{ cadraj la nivel de bit*.} \end{cases}$$

Operaţii admise :

— comparaţii numerice efectuate cu :
- valori numerice imediate ;
- caracteristici numerice ;
- variabile de lucru numerice ;

— operanzi indirecţi în calcule aritmetice (sînt operanzi indirecţi deoarece, înaintea efectuării oricăror calcule, acest tip de date trebuiesc transferate în variabile de lucru numerice care constituie operanzii expresiilor aritmetice) ;

— teste de existenţă.

Observaţie :

Este evident faptul că orice interval de variaţie, generator al unei plaje de variaţie pv, poate fi înlocuit cu un alt interval, care generează aceeaşi plajă de variaţie pv (acest lucru nu are repercusiuni asupra dimensiunii spaţiului ocupat), cu condiţia ca datele reale din fişier să se încadreze în noile limite. Acest lucru este posibil dacă în evoluţia în timp a datelor reale se constată apariţia unei diferenţe importante între datele reale şi una din margini, diferenţă care sugerează o translaţie a intervalului de variaţie.

Exemple :

AN DE 1921 A 2100 → 2100 – 1921 = 179 NBnec = 8 biţi
ZILE DE 0 A 31 → 31 – 0 = 31 NBnec = 5 biţi
LUNA DE 1 A 12 → 12 – 1 = 11 NBnec = 4 biţi
TEMPERATURA DE – 80 A + 80 (100)
| 100 – (–80)| = 100 + 80 = 180 NBnec = 8 biţi
$2^8 - 1 = 255 > 180$
VALORI-NEGATIVE DE –25 A – 20
| (–50) – (–20)| = 30Bnec NB = 5 biţi
BIT DE 0 A 0 → 0 – 0 = 0 NBnec = 1 bit
pentru a putea face distincţie între valoarea nedefinit şi valoarea 0.
AN DE 1921 A 2000 AVEC CLE FIN
COD-LOCALITATE DE 1000011 A 64999999 AVEC CLE UNIQUE FIN

Caracteristica de tip ⟨text⟩ (TEXTE)

Rol : descrierea atributelor (cîmpurilor) alfanumerice de tip text editabil;
Sintaxa :
⟨*identificator*⟩ TEXTE [(]⟨n_1⟩[⟨n_2⟩]][)]
unde :
⟨n_1⟩ $\in$ [1,255] $\cap$ N şi ⟨n_2⟩ $\in$ [1,60] $\cap$ N
Texte : defineşte un şir oarecare de caractere care se poate întinde pe mai multe linii ;
⟨n_1⟩ : defineşte numărul maxim de linii ale textului ;
⟨n_2⟩ : defineşte numărul maxim de caractere admis pentru o linie din text. Acest număr este prefixat pentru FELIX C-256/512/1024/5000 la valoarea 60.

Date acceptate : orice şir de caractere alfanumerice cu lungimea $\leqslant n_2$;

Condiţii de validare : nu există restricţii decît asupra utilizării caracterului '/' şi '*' care reprezintă caractere de control ale „editorului de texte". Textul poate fi compus din orice caractere diferite de cele două, şi trebuie să se încadreze în numărul de linii declarate. Orice tentativă de depăşire a acestui număr este semnalată ca eroare pe dispozitivul de iesire.

Spaţiul ocupat : $(15 * n_1 + 1)$ cuvinte sau m : = $(60 * n_1 + 4)$ octeţi.

Structura şi semnificaţia spaţiului ocupat :

– cuvîntul 1 :

a) semicuvîntul din stînga conţine numărul ⟨n_1⟩ de linii declarate ;

b) semicuvîntul din dreapta conţine numărul de caractere admis pentru o linie (valoare 60) ;

– liniile în care sînt introduse date vor conţine codurile interne ale caracterelor care compun aceste date ;

– liniile în care nu s-au introdus date vor conţine caracterul X'00' (nedefinit) ;

Cadraj : cuvînt.

Operaţii admise : singurele operaţii care se pot efectua cu acest gen de caracteristici sînt cele asociate funcţiunilor „editorului de texte" SOCRATE (V 1.5). În V 1.6.R gama operaţiilor este mult extinsă şi va fi prezentată ulterior în capitolul 4.

Exemple :

COMENTARIU TEXTE (10 60) → 1 + 10 * 15=151 cuvinte
DATE-GENERALE TEXTE 20 → 1 + 20 * 15=301 cuvinte

Caracteristica de tip ⟨blcc⟩

Rol : regruparea sub acelaşi identificator a unor caracteristici oarecare.

Sintaxa :

⟨*identificator*⟩ DEBUT ⟨*lista-de-caracteristici*⟩ FIN

unde :

DEBUT : marchează **începutul** ⟨*listei-de-caracteristici*⟩ care aparţin blocului ;

FIN : marchează **sfîrşitul** ⟨*listei-de-caracteristici*⟩ care apartin blocului ;

⟨*lista-de-caracteristici*⟩ : : = ⟨*caracteristică*⟩/

⟨*caracteristică*⟩⟨*lista-de-caracteristici*⟩

Din modul de producere a ⟨*listei-de-caracteristici*⟩ rezultă că declaraţia de ⟨*bloc*⟩ este recursivă. Pentru recursivitatea declaraţiei de bloc se foloseşte noţiunea de *imbricare (un bloc definit în interiorul altui bloc se numeşte ⟨bloc imbricat⟩)*. Această imbricare a blocurilor permite ierarhizarea informaţiilor pe niveluri. În cadrul blocurilor este interzisă utilizarea la acelaşi nivel a aceluiaşi identificator.

Date acceptate şi condiţii de validare :

sînt asociate şi conforme tipului de caracteristici care compun lista de caracteristici.

Spaţiul ocupat : declaraţia de bloc, ca atare, nu ocupă spaţiu suplimentar de memorie.

Fie :

n — numărul de caracteristici din listă ;

sp_i — spaţiul de memorie ocupat de caracteristica i, i = 1, 2, . . .,n ;

sc_i — spaţiul de memorie necesar cadrajului informaţiei i, i = 1, 2, . . .,n ;

(*Spaţiul de cadraj* reprezintă diferenţa dintre *cadrajul cerut* de o informaţie şi *cadrajul existent*) ;

atunci spaţiul de memorie ocupat de caracteristica de tip bloc este :

$$sb = \sum_{i=1}^{n} (sp_i + sc_i)$$

Pentru determinarea spaţiului de cadraj, facem următoarele precizări :

— informaţiile elementare, care compun o structură, sînt memorate în ordinea secvenţială a definirii lor (adresa la care sînt implantate în memorie depinde de declaraţiile anterioare — acesta reprezintă cadrajul existent ⟨ce⟩ ;

— fiecare informaţie elementară cere un anumit cadraj determinat de tipul său ⟨cc⟩ ;

— incidentele de cadraj apar, în momentul în care tipul de adresă cerut ⟨cc⟩ diferă de tipul de adresă existent ⟨ce⟩, astfel (tabela 3.1) :

Tabela 3.1. **Incidente de cadraj**

cadraj existent ⟨ce⟩	cadraj cerut ⟨cc⟩	incident de cadraj
1	2	3
cuvînt	cuvînt	nu
	octet	nu
	bit	nu
octet	cuvînt	adresa este forţată la următoarea valoare multiplu de cuvînt
	octet	nu
	bit	nu

Tabela 3.1. (continuare)

1	2	3
bit	cuvînt octet bit	adresa este forțată la următoarea valoare multiplu de cuvînt adresa este forțată la următoarea valoare multiplu de octet nu

Prioritatea de stabilire a tipului de adresă este CUVÎNT, OCTET, BIT.

Spațiul de cadraj (sc_i) cerut de o caracteristică, este reprezentat de diferența existentă între tipul de adresă cerut și tipul de adresă existentă:

$$sc_i = cc_i - ce_i$$

Deoarece ⟨cc⟩ și ⟨ce⟩ pot avea unități de măsură diferite, la detectarea unui incident de cadraj, ⟨cc⟩ va fi exprimat în unitatea de măsură a lui ⟨ce⟩ după care se efectuează calculul (sc_i are deci unitatea de măsură a lui ⟨ce⟩).

Structura și semnificația spațiului ocupat: este asociată, și conformă tipului de caracteristici cuprinse în ⟨lista-de-caracteristici⟩;

Cadraj: prima caracteristică, din ⟨lista-de-caracteristici⟩ este aliniată la CUVÎNT indiferent de tipul său; celelalte caracteristici, din bloc, sînt aliniate în conformitate cu regulile asociate tipului respectiv.

Operații admise: identificatorul unei caracteristici de tip bloc reprezintă un „tată" (în sensul organizării arborescente) pentru toate informațiile din bloc, deci poate fi utilizat drept calificator în stabilirea filiației.

Informațiile care sînt conținute în bloc (aparțin listei de caracteristici) pot fi obiectul operațiilor admise de tipul respectiv.

Exemple:

```
SITUAȚIA MILITARA
  DEBUT
  LIVRET
    DEBUT
      SERIA MOT 3
      NR    DE    0  A      999999
    FIN
    SPECIALITATE DE 0 A  999
    COMISARIAT   DE 0 A  64
    GRAD         DE 0 A  99
    OBLIGAȚII (7  1)  (1   2   3  4  5  6  7)
  FIN
DOMICILIU
  DEBUT
   STABIL
    DEBUT
     STR MOT 14
     ...
     LOCUINTA
      DEBUT
       SUPRAFATA DE 1 A 100
      FIN
    FIN
   FLOTANT
    DEBUT
```

STR MOT 14

:

FIN

FIN

Observaţie :

Identificatorul STR apare de două ori în blocul DOMICILIU. Acest lucru este admis deoarece el este definit în mod unic în blocurile imbricate STABIL şi respectiv FLOTANT. Pentru a desemna STR (strada) unuia din cele două tipuri de DOMICILIU se specifică, în limbajul de interogare, prin:

STR DE STABIL DE DOMICILIU → strada domiciliului stabil;

STR DE FLOTANT DE DOMICILIU → strada domiciliului flotant.

Caracteristica de tip ⟨entitate⟩ (ENTITE)

Rcl : descrierea unui ⟨bloc⟩ repetitiv ;

Sintaxa :

ENTITE [(]⟨*n*⟩[)]⟨*blcc*⟩

unde : ⟨*n*⟩ $\in [1,(2^{21}-1)] \cap N$

ENTITE : defineşte o caracteristică de tip ⟨*bloc*⟩ despre care se presupune că poate avea maxim ⟨*n*⟩ realizări. În acest caz identificator se numeşte „*nume de entitate*". Caracteristica de tip ⟨*entitate*⟩ păstrează proprietatea de imbricare a blocului cu menţiunea că numărul maxim de imbricări este limitat la 7.

⟨*n*⟩ : numărul maxim de realizări posibile pe care le poate avea caracteristica de tip ⟨*bloc*⟩ asociată ;

Date acceptate : asociate şi conforme tipului de caracteristici care compun entitatea.

Condiţii de validare :

— la orice tentativă de creare sau generare a unei realizări a entităţii se verifică încadrarea acesteia în numărul maxim de realizări declarate ⟨*n*⟩.

Orice tentativă de depăşire a acestui număr este semnalată ca eroare, pe suportul de ieşire, şi inefectivă ;

— pentru caracteristicile care compun entitatea, operaţiile de validare sînt conforme tipului respectiv.

Spaţiul ocupat :

— pentru caracteristica de tip bloc se calculează în conformitate cu regula dată la paragraful anterior ;

— spaţiul total ocupat de o realizare i (*st*) se calculează, în mod diferit, în funcţie de faptul că această realizare a entităţii este prima realizare a entităţii (i=1) sau diferă de aceasta.

Astfel avem :

1) $i \neq 1 : st = sb$;

2) $i = 1 : st = sb + sm$, unde ⟨*sm*⟩ este spaţiul ocupat de *două şiruri de biţi* care preced prima realizare a entităţii.

⟨*şir de biţi entitate*⟩ : : = ⟨*contor*⟩⟨*şir biţi suprimare*⟩

⟨*contor*⟩⟨*şir biţi prezenţă*⟩

⟨contor⟩ : reprezintă numărul de biţi necesari (NB nec) pentru a codifica în binar numărul maxim de valori posibile *n* (NBnec cea mai mică valoare întreagă, pozitivă, pentru care $2^{NBnec} \geqslant n$; (spaţiul real ocupat de contor se stabileşte în funcţie de numărul de biţi ocupaţi din cuvînt (ca la valori numerice) ;

⟨şir de biţi de prezenţă⟩
⟨şir de biţi de suprimare⟩ } se rezervă pentru fiecare un număr de biţi egali cu *⟨n⟩*.

Deci, fiecare şir are două părţi, partea întîi cu destinaţia de CONTOR şi partea a doua formată dintr-un număr de biţi egal cu numărul de realizări declarate în entitate :

CONTOR	1000100000000001	şirul biţilor de suprimare

CONTOR	0110001111C0110	şirul biţilor de prezenţă

1 bit pentru fiecare realizare

Spaţiul ocupat de unul din aceste şiruri (*ss*) şi contorul asociat se calculează astfel :

$$\frac{NBnec + n}{32} = Q + R \text{ dacă : } \begin{cases} R = 0, \text{ atunci } ss = Q \text{ cuvinte} \\ R \neq 0, \text{ atunci } ss = Q + 1 \text{ cuvinte} \end{cases}$$

Spaţiul total ocupat (sm) de cele două şiruri este :

$sm = 2 * ss$ cuvinte

Indiferent de numărul de realizare al entităţii aceasta trebuie să ocupe întotdeauna un număr întreg de cuvinte (dacă nu ocupă, sistemul adaugă automat informaţii de cadraj).

Pentru a optimiza accesul la realizările entităţii este indicat ca aceasta să ocupe un număr întreg de sub-pagini.

Structura şi semnificaţia spaţiului ocupat :

– *⟨contor⟩* : numărul de biţi poziţionaţi la valoarea 1 în şir ;

– *⟨şir biţi prezenţă/suprimare⟩* : bitul *k* este poziţionat la valoarea 1 dacă realizarea *k* a entităţii este, respectiv generată sau suprimată (ştearsă) ;

– pentru caracteristicile care compun blocul este asociată şi conformă, fiecărui tip.

Cadraj :

– în scopul optimizării timpilor de acces la realizările unei entităţi, este indicat ca prima realizare a entităţii să aibă o adresă multiplu de subpagină ;

– prima caracteristică din entitate este aliniată la cuvînt indiferent de tipul ei ;

– cele două şiruri de biţi sînt aliniate la cuvînt ;

– celelalte caracteristici sînt aliniate în conformitate cu regulile asociate tipului respectiv.

Operaţii admise :

⟨creare⟩/⟨ştergere⟩/⟨adăugare⟩/⟨modificare⟩ ;

– teste de existenţă.

Exemplu :

```
ENTITE 512000 PERSOANA
  DEBUT
  COD MOT 13 AVEC CLE UNIQUE FIN
  NUME MOT 16 AVEC CLE ORDONNE (26 $ ORDALFA) FIN
  PRENUME MOT 14
  DATA NASTERII
    DEBUT
     AN DE 1921 A 2100
     ZI DE 1    A   31
     LUNA DE 1 A    12
    FIN
    SEX  (2 1)  (B F)
    ENTITE 10 COPII
     DEBUT
       NUME MOT 16          /* 17 octeţi */
       PRENUME MOT 14       /* 15 octeţi */
       SEX (2 1)(M F)       /*  2 biţi */
       .
     FIN
      .
      .
      .
FIN
```

Pentru entitatea PERSOANA, spaţiul ocupat de cele două şiruri de biţi este :

— contor : 16 biţi

— şir de biţi : 512 000 deci $sm = 2 * (512\,000 + 16) = 1\,024\,032$ biţi = = 32 001 cuvinte

Caracteristica de tip ⟨inel⟩ (ANNEAU)

Rcl : descrierea relaţiilor de agregare „explicite". Permite definirea elementelor prezumtiv „posesor" în relaţiile de tip "1 — n".

Sintaxa :

⟨*identificator*⟩ ANNEAU [AVEC ⟨*tip-legătură*⟩ FIN]

⟨*tip-legătură*⟩ : : = CHAINE SIMPLE/CHAINE DOUBLE

ANNEAU : reprezintă un cap de lanţ, care leagă, pentru fiecare din realizările entităţii A în care este declarat, realizările unei entităţi R (care conţine declaraţia de referire asociată inelului) care fac referire la acestea.

Fie A (cu realizările $A_1, A_2, \ldots, A_a$) entitatea care conţine declaraţia de inel şi R (cu realizările $R_1, R_2, \ldots, R_r$) entitatea care conţine declaraţia de referire asociată. Presupunem că *realizările* R_4, R_3 şi R_1 *referă realizarea* A_1, iar realizarea R_2 referă realizarea A_2. La o declaraţie de ⟨tip-legătură⟩ = CHAINE SIMPLE, relaţia structurală introdusă în acest caz de inel şi referirea asociată se reprezintă ca în figura 3.1. unde :

— $\partial(E_i)$ reprezintă adresa realizării i din entitatea E ;

— X'00' reprezintă caracterul zero binar (nedefinit) ;

— gestiunea realizărilor grupate de acelaşi inel (gestiunea „fiilor" aceluiaşi „tată") se efectuează conform algoritmului LIFO (ultimul intrat — primul servit). În V 1.5 este materializată pe suportul real (disc) printr-o structură de tip stivă ale cărei proprietăţi sînt îmbunătăţite prin prezenţa unui cuvînt care permite reîntoar-

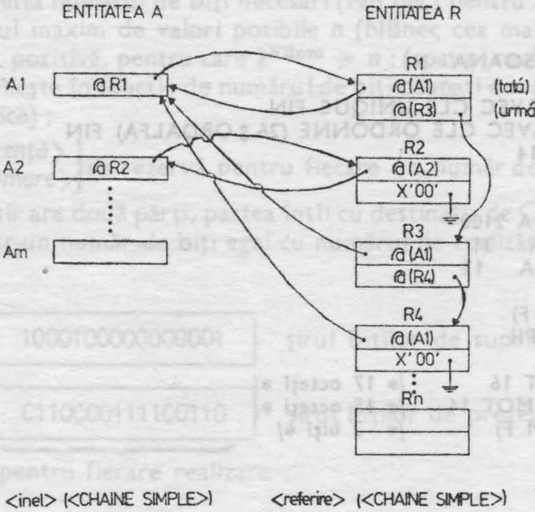


Fig. 3.1. Relația structurală introdusă de declarațiile –<referire> cu lanț simplu

cerea (punctează pe realizarea care conține inelul-„tată"), în orice moment al prelucrării, la capul de lanț („tată").

În funcție de declarația ⟨*tip-legătură*⟩ tipul de lanț introdus de caracteristica de tip ⟨*inel*⟩, poate fi:

— *simplu* (declarație: AVEC CHAINE SIMPLE FIN) — în care realizările entității care fac referire la inel sînt înlănțuite la genul: *următoarea* (acest ⟨*tip-legătură*⟩ este implicit și semnifică faptul că „tata" punctează „*ultimul său fiu*" (cel mai tînăr) iar fiecare „*fiu*" punctează spre „*fratele*" său „*mai în vîrstă*" în afară de „primul" care închide lanțul);

— *dublu* (declarație: AVEC CHAINE DOUBLE FIN) — în care realizările entității care fac referire la inel sînt înlănțuite la genul: *următoarea* și *precedenta* („*tata*" punctează spre „*primul fiu*" (cel mai în vîrstă) iar fiecare fiu punctează spre „*fratele*" său „*mai tînăr*" în afară de ultimul (cel mai tînăr) care închide lanțul).

Pentru declarația de tip ⟨*inel*⟩ se fac următoarele precizări:

— declarația de ⟨*inel*⟩ trebuie făcută totdeauna la primul nivel al entității (este interzisă introducerea lui într-un ⟨*bloc*⟩ din cadrul entității) indiferent dacă entitatea este imbricată sau nu;

— este permisă declararea mai multor caracteristici de tip ⟨*inel*⟩ în aceeași entitate;

— pentru a putea utiliza o caracteristică de tip ⟨*inel*⟩ este necesar să i se asocieze o caracteristică de tip ⟨*referire*⟩ și numai una. Dacă declarației de tip ⟨*inel*⟩ nu i se asociază o declarație de tip ⟨*referire*⟩ atunci inelul constituie un cap de lanț potențial (inutilizabil fără o eventuală asociere a unei caracteristici de tip ⟨*referire*⟩ prin adăugare la structură);

— declarația de tip ⟨*inel*⟩ și cea de tip ⟨*referire*⟩ asociată se pot face în aceeași entitate;

— declarația de tip ⟨*inel*⟩ și declarația de tip ⟨*referire*⟩ asociată trebuie făcută în același ⟨*tip-legătură*⟩;

— declarația de tip ⟨*inel*⟩, împreună cu declarația de tip ⟨*referire*⟩ asociată permite „*decuparea*" mulțimii realizărilor entității care conțin referirea în submulțimi disjuncte (permite definirea unor partiții ale datelor).

Date acceptate : adrese virtuale ale realizărilor entității care conțin declarația de tip ⟨*referire*⟩ asociată.

Condiții de validare : sînt admise numai adresele realizărilor care conțin declarația de tip ⟨*referire*⟩ asociată sau valoarea „*nedefinit*" (zero binar) care semnifică faptul că inelul nu are un lanț asociat.

Spațiul ocupat :

a) un cuvînt pentru ⟨*tip-legătură*⟩ : : = CHAINE SIMPLE ;

b) două cuvinte pentru ⟨*tip-legătură*⟩ : : = CHAINE DOUBLE.

Structura și semnificația spațiului ocupat :

a) cuvîntul de memorie, rezervat, va conține adresa ultimei realizări care a făcut referire (ultima realizare introdusă în lanț) ;

b) — primul cuvînt : idem a) ;

— al doilea cuvînt va conține adresa primei realizări care a făcut referire (cea mai veche realizare introdusă în lanț).

Cadraj : cuvînt.

Modul de acces : acces direct la realizările entității care fac referire. Accesul se face, pornind de la caracteristica de tip ⟨*inel*⟩, în conformitate cu ordinea de înlănțuire a realizărilor care fac referire (parcurgerea inelului se face secvențial iar accesul la realizare este direct).

Operații admise : comparații de adrese.

Exemple :

CANDIDATI ANNEAU AVEC CHAINE DOUBLE FIN
COPII ANNEAU
FUNCTIE ANNEAU AVEC CHAINE SIMPLE FIN.

Caracteristica de tip ⟨referire⟩ (REFERE)

Rol : descrierea relațiilor de agregare „*explicite*". Permite descrierea elementelor prezumtiv „*membru*" în relațiile de tip „1—n" sau „1—1".

Sintaxa :

⟨*referire*⟩ : : = ⟨*referire simplă*⟩/⟨*referire cu inel*⟩
⟨*referire simplă*⟩ : : = ⟨*identificator*⟩ REFERE {UN/UNE}/⟨*nume entitate*⟩
[⟨*calificare*⟩]
⟨*referire cu inel*⟩ : : = ⟨*identificator*⟩ REFERE ⟨*nume inel*⟩
DE {UN/UNE} ⟨*nume entitate*⟩[⟨*calificare*⟩]
[AVEC ⟨*tip-legătură*⟩ FIN]
⟨*nume entitate*⟩ : : identificatorul entității la care se face referire ;
⟨*calificare*⟩ : : = DE {UN/UNE} ⟨*nume entitate*$_1$⟩ [⟨*calificare*⟩]/
DE ⟨*nume blocaj*⟩ [⟨*calificare*⟩] DE {UN/UNE} ⟨*nume entitate*⟩
⟨*tip legătură*⟩ : : = CHAINE SIMPLE/CHAINE DOUBLE
REFERE : definește o relație structurală de filiație ;
⟨*nume inel*⟩ : identificatorul caracteristicii de tip ⟨*inel*⟩ căreia i se asociază ;

⟨*nume entitate*$_i$⟩ ⟨*nume blocaj*⟩ } : identificatorii caracteristicilor de tip ⟨*bloc*⟩ şi/sau ⟨*entitate*⟩ în care este definită entitatea la care se face referire. Ordinea de declarare a acestor elemente de calificare este dată de modul de definire a filiaţiei la descrierea structurilor imbricate (descrierea structurilor imbricate permite definirea unor relaţii de agregare „implicite" care modelează structurile arborescente).

UN/UNE : precizează faptul că identificatorul care urmează este un nume de entitate. Cele două forme sînt echivalente şi permit efectuarea acordului gramatical între elementele descrise (apropierea limbajului utilizat de limbajele naturale).

Relaţia structurală, introdusă de caracteristica de tip ⟨*referire*⟩ depinde de tipul său şi de ⟨*tip-legătură*⟩, astfel :

— pentru caracteristicile de tip ⟨*referire cu inel*⟩ — lanţ simplu — se prezintă conform celor arătate anterior (figura 3.1) ;

— pentru caracteristicile de tip ⟨*referire cu inel*⟩ — lanţ dublu — se prezintă ca în figura 3.2 (luăm cazul R_4, R_3, R_1 referă A_1) ;

— pentru caracteristicile de tip ⟨*referire simplă*⟩, se prezintă ca în figura 3.3.

Date acceptate : adrese virtuale ale entităţii referite.

Condiţii de validare : sînt admise numai adresele virtuale ale realizărilor entităţii care conţin declaraţia de inel (⟨*referire cu inel*⟩) sau numai adresele virtuale ale realizărilor entităţii referite (⟨*referire simplă*⟩).

Spaţiul ocupat : depinde de tipul declaraţiei astfel :

1) ⟨*referire simplă*⟩ : 1 cuvînt ;
2) ⟨*referire cu inel*⟩ — lanţ simplu : 2 cuvinte ;
3) ⟨*referire cu inel*⟩ — lanţ dublu : 3 cuvinte.

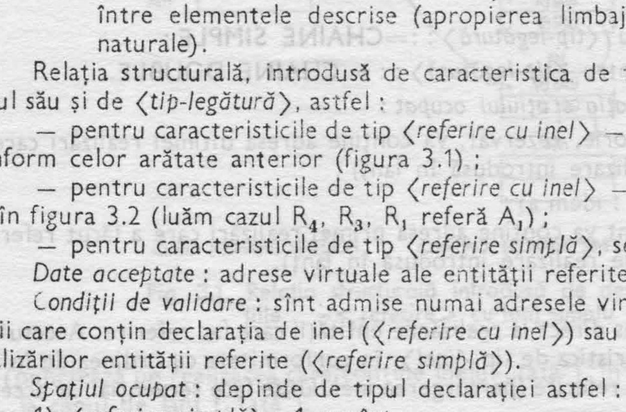


Fig. 3.2. Relaţia structurală introdusă de declaraţiile ⟨ inel ⟩ — ⟨ referire ⟩ cu lanţ dublu

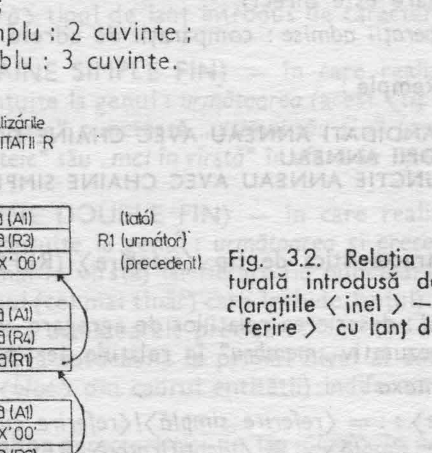


Fig. 3.3. Relaţia structurală introdusă de declaraţia ⟨ referire ⟩ simplă

Structura și semnificația spațiului de memorie ocupat :

1) — cuvîntul de memorie conține adresa virtuală a realizării la care se face referire. Prin această adresă se face asocierea unilaterală entitate care face referire — entitate referită. În acest caz, dacă ne situăm în realizarea care face referire, avem acces direct la toate valorile atributelor entității referite (cuvîntul de memorie reprezintă adresa de bază a realizării iar accesul la caracteristici este făcut adunînd adresa relativă a acestora la adresa de bază).

2) — cuvîntul 1 : conține adresa virtuală a realizării entității, care conține declarația de ⟨*inel*⟩ asociată, la care se face referire — vezi 1) ;

— cuvîntul 2 : conține adresa virtuală a următoarei realizări a entității care face referire la aceeași realizare a entității care conține declarația de ⟨*inel*⟩ asociată. Acest cuvînt permite înlănțuirea la modul „următorul" frate. Algoritmul de gestiune al acestei stive este de tip LIFO (ultimul intrat — primul servit). Ultima realizare din lanț (prima introdusă în mod cronologic) conține valoarea zero binar (nedefinit) ;

3) — cuvîntul 1 și 2 : identic 2) ;

— cuvîntul 3 : conține adresa virtuală a precedentei realizări (a entității care conține declarația ⟨*referire cu inel*⟩) care a făcut referire la aceeași realizare a entității care conține declarația de ⟨*inel*⟩ asociată.

Acest cuvînt permite înlănțuirea la modul „precedentul" frate. Algoritmul de gestiune al acestei stive este de tip FIFO (primul intrat — primul servit). Ultima realizare din lanț (ultima introdusă în sens cronologic) conține valoarea nedefinit.

Cadraj : cuvînt.

Modul de acces : acces direct la realizările entității la care se face referire (inclusiv orice atribut conținut de acestea). Pentru ⟨*referire cu inel*⟩ este permis, în funcție de declarația ⟨*tip legătură*⟩ și accesul la „*precedentul*" sau „*următorul*" (implicit) frate.

Operații admise : comparații de adrese.

Observații :

Practic, pe suportul real, prin declarația REFERE se va obține un tablou de indecși de dimensiune variabilă. Valorile elementelor din tablou (index) sînt adrese ale unor realizări de entitate și vor fi introduse în acesta în funcție de conținutul datelor reale, obținîndu-se astfel o alocare dinamică a tabelei. Elementele componente ale tabelei sînt dispersate în spațiul real iar recompunerea tabelei se realizează pornind de la capul de lanț (ANNEAU asociat).

Parcurgerea poate fi completă sau nu în funcție de cerințele utilizatorului specificate în limbajul de interogare. Sistemul mai oferă și posibilitatea de adresare a oricărui element din tabelă și parcurgerea tabelei din acest punct pînă la oricare punct al său astfel :

— cu REFERE — CHAINE SIMPLE — numai în sensul „*următorul*" index ;

— cu REFERE — CHAINE DOUBLE — în oricare din sensurile „*următorul*" sau „*precedentul*" fiind permisă și trecerea în orice moment dintr-un sens de parcurgere în celălalt (pentru sensul „*precedentul*" utilizatorul trebuie să-și construiască un program IMT care să realizeze accesul la tabelă).

Alegerea între o declarație de ⟨*referire cu inel*⟩ cu lanț simplu sau lanț dublu se face ținînd cont de spațiul de memorie necesar reprezentării și de numărul elementelor din lanț și frecvența ștergerilor (inserările în lanț au timpi de execuție sensibil apropiați).

Dacă n este numărul elementelor din lanț iar t necesarul de timp pentru o operație elementară (citire sau scriere) atunci timpul mediu de efectuare a unei ștergeri (ts) se calculează astfel :

— lanț simplu ; $ts = \left(\frac{n}{2} + 2\right) * t$ (crește proporțional cu n) ;

— lanț dublu : $4 * t$ (nu depinde de n).

Spaţiul de memorie ocupat este cu (n + 1) cuvinte mai mare la legătura cu lanţ dublu faţă de cea cu lanţ simplu.

Exemple:

Pentru descrierea structurii administrativ-teritoriale a R.S.R. reprezentată în figura 2.5, vom efectua următoarele declaraţii:

```
/* descriere entitate JUDET                                    */
ENTITE 64 JUDET
  DEBUT
  /* legătura „1—1" spre centrul administrativ                 */
  CENTRU-ADMINISTRATIV REFERE UNE LOCALITATE
  /* gruparea localităţilor care aparţin judeţului             */
  /* legătura de tip „1—n", lanţ dublu                         */
  LOCALITATI ANNEAU AVEC CHAINE DOUBLE FIN
  /* denumirea judeţului 40 caractere                          */
  DEN1 MOT 20
  DEN2 MOT 20
FIN
/* descriere entitate LOCALITATE                               */
ENTITE 10240 LOCALITATE
  DEBUT
  /* judeţul de care aparţine                                  */
  JUDET REFERE LOCALITATI DE UN JUDET
              AVEC CHAINE DOUBLE FIN
  /* persoanele născute în localitate — pointer potenţial      */
  NASCUTI ANNEAU AVEC CHAINE SIMPLE FIN
  /* localităţi subordonate administrativ                      */
  LOC-SUP REFERE LOC-SUB
     DE UN LOCALITATE
  /* codul localităţii ; cheie de acces unică                  */
  COD DE 1 A 99999 AVEC CLE UNIQUE FIN
  /* denumirea localităţii 40 caractere:                       */
  /* primele 20 caractere, declarate cheie de acces,           */
  /* permit parcurgerea în ordine strict alfabetică, dacă      */
  /* este nevoie.                                              */
  DEN1 MOT 20 AVEC CLE (26 $ORDALFA) FIN
  DEN2 MOT 20
FIN
/* ORDALFA : program „hashing" construit de                    */
/* utilizator */
```

Caracteristica de tip ⟨invers⟩ (INVERSE)

Rol: definirea unei strategii de acces la realizările unei entităţi, selecţionate conform unui „criteriu de selecţie" specificat de utilizator. Criteriul de selecţie reprezintă o condiţie, simplă sau compusă, aplicată valorilor atribuite caracteristicilor care aparţin entităţii.

Sintaxa:

⟨*identificator*⟩ INVERSE {TOUT/TOUTE} ⟨*nume entitate*⟩
[⟨*calificare*⟩]

⟨*calificare*⟩ : : = DE {TOUT/TOUTE} ⟨*nume entitate*$_i$⟩ [⟨*calificare*⟩]
DE {UN/UNE} ⟨*nume entitate*$_r$⟩/
DE ⟨*nume bloc*$_j$ [⟨*calificare*⟩] DE {UN/UNE}
⟨*nume entitate*$_r$⟩

⟨*nume entitate*⟩ : identificatorul entităţii căreia îi asociem caracteristica de tip ⟨*invers*⟩ ;

UN/UNE : cuantificatori precizînd faptul că citaţia de calificare se referă la o singură realizare de entitate ;

TOUT/TOUTE : cuantificatori precizînd faptul că citaţia de calificare se referă la toate realizările entităţii.

INVERSE : defineşte un ⟨*şir de biţi*⟩ (analog şirului de biţi de prezenţă), a cărui lungime este determinată de imbricarea entităţii pe care o inversăm. Numărul biţilor din listă este egal cu produsul numerelor maxime de realizări posibile ale entităţilor cuantificate prin TOUT/TOUTE, care *califică* entitatea inversată. În cazul entităţilor imbricate, sfîrşitul citaţiei de inversare se cuantifică prin UN/UNE şi determină realizarea entităţii (nume entitate$_i$) pentru care se generează realizările entităţii inversate ;

⟨*nume entitate*$_i$⟩
⟨*nume bloc*$_j$⟩ } : identificatorii caracteristicilor de tip ⟨*bloc*⟩ şi/sau ⟨*entitate*⟩ în care este definită entitatea pe care o inversăm. Ordinea de declarare a acestor elemente este dată de modul de definire a filiaţiei la descrierea structurilor imbricate.

Date acceptate : bitul i din şir ia valoarea 1 dacă realizarea corespunzătoare a entităţii este inversată (dacă satisface criteriul stabilit de utilizator) altfel are valoarea 0 (zero).

Condiţii de validare : este acceptată numai inversarea realizărilor entităţii care corespund citaţiei de stabilire a filiaţiei.

Spaţiul ocupat : ⟨*inel*⟩ ⟨*contor*⟩ ⟨*şir de biţi*⟩

⟨*inel*⟩ : 1 cuvînt ;

⟨*contor*⟩ : numărul de biţi necesari (NBnec) pentru a codifica în binar numărul maxim de realizări posibile (Nmr) pe care le poate avea entitatea inversată. Dacă $n_1, n_2, \ldots, n_i$ $(i = 1,7)$ sînt numerele maxime de realizări posibile ale entităţilor imbricate care conţin entitatea inversată i, atunci :

$\mathrm{Nmr} = n_1 * n_2 * \ldots * n_i = \prod_{i=1}^{p} n_i,\ 1 \leqslant p \leqslant 7$, iar NBnec reprezintă cea mai mică valoare întreagă pentru care $2^{\mathrm{NBnec}} > \mathrm{Nmr}$;

⟨*şir de biţi*⟩ : se rezervă un şir de Nmr biţi.

Spaţiul de memorie ocupat (*sm*) de şirul de biţi şi contor se determină astfel :

$$\frac{\mathrm{Nmr} + \mathrm{NBnec}}{32} = Q + R \text{ (32 numărul de biţi dintr-un cuvînt de memorie)},$$

$$\text{dacă } R = \begin{cases} 0, \text{ atunci } sm = Q \text{ cuvinte ;} \\ 1, \text{ atunci } sm = Q + 1 \text{ cuvinte.} \end{cases}$$

Spaţiul total ocupat este de $(sm + 1)$ cuvinte (1 cuvînt pentru ⟨*inel*⟩).

Structura şi semnificaţia spaţiului de memorie ocupat :

⟨*inel*⟩ : conţine adresa virtuală de origine a primei realizări a entităţii inversate ;

⟨*contor*⟩ : numărul de biţi poziţionaţi la valoarea 1 în şir ;

⟨*şir de biţi*⟩ : bitul i are valoarea 1 dacă realizarea i a entităţii a făcut obiectul unei comenzi de inversare (realizarea este considerată în contextul ierarhiei stabilite la declarare).

Cadraj : cuvînt.

Modul de acces : acces direct la realizările entităţii inversate.

Operaţii admise : comparaţii de adrese, teste de existenţă a unei realizări inversate şi criteriu de acces la datele din baza de date.

Observații :

Deși datele acceptate la gruparea pe o caracteristică de tip ⟨*invers*⟩ sînt reprezentate de adrese virtuale ale entității căreia i se asociază, semantica grupajelor poate fi schimbată de utilizator. Pentru acest mod de utilizare se oferă mai multe informații în capitolul 10.

Exemple :

```
1) ENTITE 1000 PERSOANA
   DEBUT
   MARCA DE 1111 A 9999
   NUME MOT
   SEX (2  1) (B  F)
   PROFESIE MOT 25
   FIN
   BARBAŢI INVERSE TOUT PERSOANA
   ECONOMISTI INVERSE TOUT PERSOANA
```

Nmr = 1000

NBnec = 10 biți ($2^{10} = 1024 > 1000$)

$$\frac{Nmr + NBnec}{32} = \frac{1000 + 10}{32} = 31 \text{ rest, } 18 \text{ deci } sm = 32 \text{ cuvinte}$$

Spațiul total ocupat este 33 cuvinte.

Pentru datele prezentate în tabela următoare se generează în fișierele inverse configurațiile de biți prezentate mai jos (tabela 3.2).

Tabela 3.2 Fișier obiect ⟨PERSOANA⟩

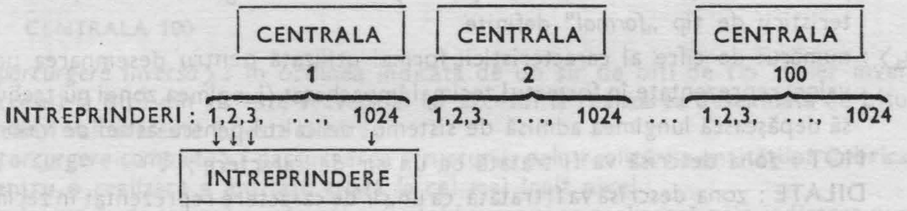
Număr realizare	MARCA	NUME	SEX	PROFESIE
1	1111	IONESCU	F	ECONOMIST
2	1211	POPESCU	M	INGINER
3	1113	DANILOV	M	ECONOMIST
4	1214	RADUCAN	M	MEDIC
5	1315	VASILE	M	MUNCITOR
6	1126	SANDU	F	MUNCITOR
7	1173	DORU	M	ECONOMIST
.				
.				
1 000				

FIȘIERE INVERSE

Caracteristica de tip ⟨invers⟩	Condiția de inversare	Număr bit poziționat								
		1	2	3	4	5	6	7	...	1 000
BARBATI	SEX='M'	0	1	1	1	1	0	1		
ECONOMISTI	PROFESIA= ECONOMIST	1	0	1	0	0	0	1		

```
2) ENTITE 10 MINISTER
     DEBUT
       INTREPRINDERI INVERSE TOUT INTREPRINDERE
                             DE TOUT CENTRALA
                             DE UN MINISTER
       ENTITE 100 CENTRALA
         DEBUT
           :
           ENTITE 1024 INTREPRINDERE
             DEBUT
               :
               :
             FIN
         FIN
FIN
```

Nmr=1024 * 100=102400 biţi
NBnec=17 biţi ($2^{10} * 2^{7}$=1024 * 128)
102417 : 32=3200 rest 17, deci *sm*=3201 cuvinte
Spaţiul total ocupat este 3202 cuvinte
Structura semantică a şirului de biţi, pentru un minister, se prezintă astfel :



Caracteristică de tip ⟨formal⟩ (FORMAL)

Rol : descrierea structurii fişierelor externe asociate bazei de date. Aceste fişiere conţin datele de intrare/ieşire necesare aplicaţiilor funcţionale care concură la exploatarea ⟨*bazei de date*⟩.

Caracteristicile de tip ⟨*formal*⟩ reprezintă caracteristici „fictive", adică, pentru ele, nu se rezervă spaţiu apriori nici în baza de date nici în memoria centrală. Aceste caracteristici permit structurarea unui buffer de intrare/ieşire conform reprezentării arborescente a unei înregistrări de tipul celor utilizate în limbajul COBOL. Această structurare reprezintă de fapt o redefinire, la momentul execuţiei (prin citaţii specifice), a acestei zone de memorie.

Identificatorii caracteristicilor definite în ⟨formal⟩ permit adresarea acestor zone redefinite iar tipul asociat lor specifică modul de structurare şi natura datelor (deci modul specific de tratare în cazul operaţiilor efectuate cu aceste zone).

Sintaxa :

⟨*formal logic*⟩ : : = FORMAL ⟨*bloc formal*⟩
⟨*bloc formal*⟩ : : = ⟨*identificator*⟩ DEBUT ⟨*lista de caracteristici formal*⟩ FIN
⟨*lista de caracteristici formal*⟩ : : = ⟨*caracteristică formal*⟩/
⟨*caracteristică formal*⟩ ⟨*lista de caracteristici formal*⟩
⟨*caracteristică formal*⟩ : : = ⟨*cuvînt formal*⟩/⟨*zecimal despachetat*⟩/
⟨*zecimal împachetat*⟩/⟨*formal logic repetitiv*⟩

⟨*cuvînt formal*⟩ : : = ⟨*identificator*⟩ MOT [(][⟨n_1⟩] ⟨n_2⟩ [)]
⟨*zecimal despachetat*⟩ : : = ⟨*identificator*⟩ DILATE [(][n_1] n_2 [)]
⟨*zecimal împachetat*⟩ : : = ⟨*identificator*⟩ PACKE [(] ⟨n_3⟩ [)]
⟨*formal logic repetitiv*⟩ : : = FORMAL [(] *n* [)] ⟨*bloc formal*⟩
unde : ⟨n⟩ $\in [1,(2^{24}-1)] \cap N$, ⟨$n_2$⟩ $\in [1,80] \cap N$ $n_1 \in [1,$ ⟨*l. form*⟩]

⟨n⟩ : indică factorul maxim de repetare al unui formal logic repetitiv în interiorul altui ⟨*formal logic*⟩ sau ⟨*formal logic repetitiv*⟩. Un ⟨*formal logic repetitiv*⟩ permite descrierea unor matrice (numărul de dimensiuni este egal cu numărul de imbricări a caracteristicilor de tip ⟨*formal logic repetitiv*⟩), permițînd exploatarea structurilor definite în limbajele de nivel înalt (articole definite cu clauza OCCURS în COBOL sau DIMENSION în FORTRAN).

⟨n_2⟩ : indică lungimea zonei de memorie adresată de ⟨*identificator*⟩ (octeti) ;

⟨n_1⟩ : definește poziția de început a zonei de memorie adresată de identificator. Dacă ⟨n_1⟩ nu este indicat atunci începutul zonei adresate de ⟨*identificator*⟩ este dat de poziția în care s-a ajuns prin definirile anterioare +1. Variabila ⟨n_1⟩ servește, în esență, la redefinirea unor caracteristici ⟨*formal*⟩ (redefinirea nu are sens decît asupra unor caracteristici definite anterior, adică valoarea ($n_1 + n_2 - 1$) nu trebuie să depășească valoarea poziției la care s-a ajuns prin definirile anterioare). Prin ⟨*l form*⟩ s-a notat lungimea în octeți a caracteristicii de tip „*formal*" definite.

⟨n_3⟩ : numărul de cifre al caracteristicii formal utilizată pentru desemnarea unei valori reprezentate în formatul zecimal împachetat (lungimea zonei nu trebuie să depășească lungimea admisă de sistemul de calcul pentru astfel de valori).

MOT : zona descrisă va fi tratată ca un șir de caractere ;

DILATE : zona descrisă va fi tratată ca un șir de caractere reprezentat în zecimal despachetat (fiecare caracter este de exemplu de forma X'F ⟨*cifră*⟩' în afară de ultimul care poate conține în locul lui X'F.⟨*cifră*⟩', X'C ⟨*cifră*⟩' pentru numere pozitive sau X'D ⟨cifră⟩' pentru numere negative cu ⟨cifră⟩ : : = 0/1/2/3/4/5/6/7/8/9).

PACKE : zona descrisă va fi tratată ca un șir de caractere reprezentat în zecimal împachetat (fiecare caracter este de exemplu de forma X' ⟨*cifră*⟩ ⟨*cifră*⟩, în afară de ultimul care poate fi X' ⟨*cifră*⟩ F' sau X' ⟨*cifră*⟩ C' pentru numere pozitive sau X' ⟨*cifră*⟩ D' pentru numere negative).

Observație : utilizarea caracteristicilor de tip ⟨*formal*⟩ nu impune folosirea unor fișiere externe. Programatorul de aplicație poate utiliza aceste caracteristici pentru a structura un buffer de memorie centrală conform necesităților sale de prelucrare.

Exemplu :

Dacă dorim să reprezentăm o MATRICE cu 10 LINII și 100 de COLOANE, avînd valori zecimale despachetate se realizează astfel :

```
FORMAL MATRICE
DEBUT
  FORMAL 10 LINIA
  DEBUT
    FORMAL 100 COLOANA
    DEBUT
      VALOARE DILATE 10
    FIN
  FIN
FIN
```

Utilizarea elementului <VALOARE> solicită specificarea filiației sale în cadrul structurii descrise, la modul (vezi limbajul de manipulare a datelor) :
VALOARE DE FORMAL COLOANA K DE FORMAL LINIA 1 DE FORMAL MATRICE
unde K reprezintă numărul coloanei iar I numărul liniei în care se află elementul care interesează. Pentru desemnarea unui element se poate utiliza și forme mult mai concise de exprimare pe care cititorul le va regăsi în capitolul 4.

Exemple :

1) Descrierea structurii unor machete de tip cartelă perforată :

```
FORMAL CARTELA
  DEBUT
  IDENT   MOT 3              /* zonă de 3 caractere alfanumerice */
  NR      DILATE (2)         /* zonă de 2 caractere zecimal despachetate  */
  FILLER MOT 75              /* zonă de 75 caractere alfanumerice */
  COD DILATE 5 5             /* redefinirea zonei FILER:                  */
  NUME MOT (10 14)           /* COD-zecimal despachetat, 5 poziții        */
  PRENUME MOT (24 10)        /* NUME, PRENUME-șiruri alfanumerice         */
FIN
```

Machetele înregistrărilor care corespund descrierii formalului CARTELA sînt :

IDENT	NR	FILLER
		X(75)
1 3	4 5	6

IDENT		COD	NUME	PRENUME	
X(3)		9(5)	X(14)	2X(10)	
1 3	4	5 9	10 23	24 33	34

2) Descrierea structurii unui fișier pe suport magnetic

```
FORMAL BANDA
  DEBUT
  COD PACKE 5
  FORMAL 4 TRIMESTRE
    DEBUT
     FORMAL 3 LUNI
       DEBUT
         VALOARE DILATE 6
         CANTITATE DILATE
       FIN
    FIN
FIN
```

Înregistrarea corespunzînd acestei descrieri are structura (tab. 3.3):

Tabela 3.3. **Structura înregistrării**

COD P(5)	TRIMESTRUL 1						TRIMESTRUL 2	
	LUNA 1		LUNA 2		LUNA 3		LUNA 1	
	Valoare 9(6)	Cantitate 9(3)	Valoare 9(6)	Cantitate 9(3)	Valoare 9(6)	Cantitate 9(3)	Valoare 9(6)	Cantitate 9(3)

←— de 4 ori —→

Lungimea înregistrării definite este $RCS = 5 + 4 * 3 * (6 + 3) = 113$ octeți

Tipuri de declarații de acces

⟨*tip acces*⟩ : : = ⟨*secvențial*⟩/⟨*direct*⟩/⟨*secvențial indexat*⟩
⟨*secvențial*⟩ : : = ⟨*parcurgere naturală*⟩/⟨*parcurgere inversă*⟩/⟨*parcurgere completă*⟩
⟨*direct*⟩ : : = ⟨*prin număr de ordine*⟩/⟨*prin dicționar*⟩
⟨*parcurgere naturală*⟩ : în ordinea naturală de prezentare a realizărilor entităților conform metodei de organizare dispersată;

Metoda de organizare dispersată permite gruparea tuturor realizărilor unei entități într-un spațiu virtual continuu chiar dacă modul de memorare pe suportul extern este discontinuu. Referindu-ne la exemplul 2) din paragraful precedent vom avea o reprezentare de forma:

MINISTER 1
  INTREPRINDERI
  CENTRALA 1
  .
  .
    INTREPRINDERE 1
    .
    .
    .
    INTREPRINDERE 2
    .
    .
    INTREPRINDERE 3
    .
    .
    . . . .
    INTREPRINDERE 1024
    .
    .
    .
  CENTRALA 2
    INTREPRINDERE 1
    .
    .

În această formă de prezentare fiecare realizare a entității MINISTER grupează toate realizările entității CENTRALA care îi aparțin iar fiecare realizare a entității CENTRALA regrupează toate realizările entității INTREPRINDERE.

```
INTREPRINDERE 2
      .
      .
- - - - - - - - - - - - - - -
      .
      .
      .
CENTRALA 100
.
.
.
MINISTER 2
  INTREPRINDERI
    CENTRALA 1
      .
      .
      .
- - - - - - - - - - - - - - - - -
MINISTER 10
  INTREPRINDERI
  CENTRALA 1
  .
  .
  .
  CENTRALA 100
```

⟨*parcurgere inversă*⟩ : în ordinea indicată de un şir de biţi de tip „fişier invers". Accesul la biţii din şir este secvenţial iar accesul la realizarea desemnată de bitul i, a cărui valoare este 1, este direct ;

⟨*parcurgere completă*⟩ : parcurgerea structurii, prin realizările entităţilor imbricate, pentru o realizare a entităţii aflată la cel mai înalt nivel ;

⟨*prin număr de ordine*⟩ : este cel mai rapid acces şi se realizează prin indicarea numărului realizării căutate (dacă realizarea aparţine unei entităţi imbricate, aflată la un nivel inferior, este necesară specificarea numerelor de ordine a entităţilor care o includ, conform imbricării) ;

⟨*prin dicţionar*⟩ : parcurgerea punctuală a realizărilor entităţii după o caracteristică declarată cheie de acces. Parcurgerea se efectuează prin aplicarea unei funcţii de „hash-cod" (dispersare) valorii de căutare, funcţie care găseşte o intrare într-un dicţionar care conţine adresa virtuală a realizării căutate ;

⟨*secvenţial indexat*⟩ : se efectuează parcurgînd secvenţial o tabelă de indecşi. Fiecare intrare din tabela de indecşi ne trimite la realizarea asociată prin adresa virtuală a acesteia.

Pentru ca o caracteristică din baza de date să permită modul de acces ⟨*direct*⟩ sau ⟨*indexat secvenţial*⟩ este necesar să se specifice acest lucru, la definire, prin ⟨*declaraţie acces*⟩. Specificarea ⟨*declaraţiei de acces*⟩ transformă caracteristica respectivă în ⟨*cheie de acces*⟩. Declararea cheilor de acces trebuie să fie făcută la nivelul 1 al entităţii (din punct de vedere al structurii arborescente a acesteia). Nu există nici o restricţie asupra numărului de caracteristici declarate ⟨*chei de acces*⟩ în aceeaşi entitate. Totuşi, utilizatorul, trebuie să limiteze numărul de chei, la cele strict necesare, deoarece pentru fiecare realizare a cheii se generează o intrare într-o tabelă de indecşi (deci spaţiu de memorie externă ocupat în plus).

Declararea cheilor de acces se face în funcţie de tipul caracteristicii la care se asociază astfel :

1) pentru ⟨*lista de valori*⟩ :

$$\langle \textit{declarație acces}_2 \rangle ::= \text{AVEC CLE} \left[\begin{Bmatrix} \text{UNIQUE} \\ \text{DOUBLE AVANT} \\ \text{DOUBLE ARRIERE} \end{Bmatrix} \right]$$

$$\left[\begin{Bmatrix} \text{CHAINE SIMPLE} \\ \text{CHAINE DOUBLE} \end{Bmatrix} \right] \text{FIN}$$

2) pentru ⟨*cuvînt*⟩ şi ⟨*valoare numerică*⟩ :

$$\langle \text{declarație acces}_1 \rangle ::= \text{AVEC CLE} \left[\begin{Bmatrix} \text{UNIQUE} \\ \text{DOUBLE AVANT} \\ \text{DOUBLE ARRIERE} \end{Bmatrix} \right]$$

$$\left[\begin{Bmatrix} \text{CHAINE SIMPLE} \\ \text{CHAINE DUBLE} \end{Bmatrix} \right] \left[\begin{Bmatrix} \text{ORDONNE} \\ \text{NON ORDONNE} \end{Bmatrix} \right] [([\langle \textit{index} \rangle]\ [\$ \langle \textit{prog} \rangle])]$$

$$\text{FIN}$$

⟨*cheie de acces*⟩ : : = ⟨*discriminantă*⟩/⟨*rapidă*⟩

⟨*discriminantă*⟩ : în acest caz nu putem să avem valori identice pentru două sau mai multe realizări ale caracteristicii (*CLE UNIQUE*) ;

⟨*rapidă*⟩ : în acest caz sînt admise mai multe valori identice pentru caracteristica declarată cheie (CLE DOUBLE AVANT, CLE DOUBLE ARRIERE).

Modul de aranjare a intrărilor din tabela de indecşi (deci modul de acces la realizările entităţii) pentru caracteristicile ⟨*chei de acces rapide*⟩ depinde de opţiunea de declarare :

– DOUBLE AVANT : ultima realizare este aranjată înaintea celorlalte (se aplică algoritmul LIFO „ultimul intrat – primul servit") ;

– DOUBLE ARRIERE (opţiune implicită) : ultima realizare introdusă este aranjată după cele definite anterior (se aplică algoritmul LILO „ultimul intrat – ultimul servit").

Deoarece declararea unei ⟨*chei de acces*⟩ ⟨*rapidă*⟩ admite apariţia de sinonime (se obţine aceeaşi valoare prin aplicarea funcţiei de „hash-cod"), avem următoarele opţiuni de înlănţuire a acestora (vezi caracteristicile de tip ⟨*inel*⟩ şi ⟨*referire*⟩) :

– CHAINE SIMPLE : sinonimele sînt înlănţuite la modul *următorul* index (implicit) ;

– CHAINE DOUBLE : sinonimele sînt înlănţuite la modul *următorul* şi *precedentul* index. În versiunea SOCRATE V 1.5. această opţiune este admisă de compilatorul LDD dar este inefectivă ;

– ORDONNE : sinonimele sînt aranjate în ordinea crescătoare a cheilor ;

– NON ORDONNE : sinonimele sînt aranjate în ordinea naturală a sosirii lor (opţiune implicită).

Alegerea unei opţiuni diferite de cea implicită va provoca o creştere a timpului de prelucrare al acestui tip de caracteristici (la operaţiile de creare/ştergere/modificare), de aceea administratorul bazei de date trebuie să decidă numărul minim de caracteristici necesare cu aceste opţiuni.

⟨*index*⟩ : defineşte numărul de intrări distincte în dicţionarul asociat caracteristicii, ⟨*index*⟩ $\in [1, (2^{16} - 1)] \cap N$. Funcţia de „hash-cod" aplicată valorii caracteristicii trebuie să furnizeze un număr în intervalul [1, ⟨*index*⟩]. Dacă index nu este specificat atunci sistemul îi stabileşte valoarea pentru valori numerice în funcţie de „plaja de variaţie a valorii numerice" sau în funcţie de imbricarea entităţii care conţine această caracteristică (declarată totdeauna la primul nivel al entităţii). Pentru caracteristicile de tip ⟨cuvînt⟩ programul de „hash-cod", furnizat de sistem, este nonclasant. În cazul în care utilizatorul doreşte aranjarea elementelor din tabela de indecşi, pentru aceste caracteristici, într-o anumită ordine este necesară construirea unui program de „hash-cod" propriu.

Pentru caracteristicile de tip ⟨listă de valori⟩ nu este necesară declararea acestui element deoarece lista în sine constituie un dicţionar cu un număr prestabilit de intrări.

⟨*prog*⟩ : este numele unui program de „hash-cod", scris în ASSIRIS, memorat în spaţiul PROG al bazei de date. Acest program, analizînd valoarea atribuită caracteristicii în cauză, trebuie să furnizeze o valoare în intervalul [1, ⟨*index*⟩].

Exemple :

```
ENTITE 32 PROFESIA
  DEBUT
    PRO-PER ANNEAU
        AVEC CHAINE DOUBLE FIN
    /* ⟨lista de valori⟩ : cheie unică pentru regăsirea unei realizări PROFESIA */
    COD-PRO (32 1) (1 2 3 4 5 6 7 8 9
                    A B X)
        AVEC CLE UNIQUE FIN
  NUME MOT
FIN
ENTITE 512000 PERSOANA
  DEBUT
    PER-PRO REFERE PRO-PER DE UN PROFESIA
        AVEC CHAINE DOUBLE FIN
    /* COD-PER : cheie unică de regăsire a unei PERSOANA */
    /* algoritmul de hash-cod este cel furnizat de sistem */
    COD-PER MOT 13 AVEC CLE UNIQUE FIN
    /* COD-PER ::= ⟨AS,A,LL, ZZ,NNNNN,C⟩ */
    /* Pentru gruparea persoanelor în ordine strict crescătoare */
    /* a ⟨numelui⟩ se foloseşte programul de hash-cod ⟨ORDALFA⟩ */
      NUME MOT 30 AVEC CLE ORDONNE (29 $ORDALFA) FIN
    /* caracteristica este cheie de acces discriminantă */
    /* sinonimele fiind reprezentate de aceeaşi rădăcină */
    /* pentru ⟨nume⟩                                      */
FIN
```

Compilarea textelor de definire

Faza de compilare a descrierii structurii constă în traducerea textului, scris în limbaj sursă, într-o formă internă codificată. Această formă internă va constitui, dacă este corectă, dicţionarul bazei de date.

Acest dicţionar este memorat în baza de date, într-un spaţiu rezervat (denumit în general STR...), pentru a constitui modelul de referinţă pentru orice utilizare ulterioară a bazei de date (creare, interogare, actualizare etc.).

Forma internă obţinută are următoarele calităţi:

— este *simplă*, adică poate fi interpretată uşor şi rapid ;

— este *densă*, adică conţine maximul de informaţii despre noţiunile care apar în textul sursă precum şi legăturile care există între ele ;

— este *completă*, adică poate fi recompusă (plecînd de la codul intern se poate obţine codul sursă care l-a generat) ;

— este *relativ simplă*, adică, în anumite condiţii, poate fi modificată fără a afecta organizarea datelor pe suportul real.

Construirea dicţionarului

Obiectele pe care le manipulează compilatorul de structură sînt: entităţi, blocuri, caracteristici simple. Datorită faptului că scrierea textului sursă se face în format liber, traducerea sa necesită existenţa unui analizor lexical care să recunoască în text fiecare unitate lexicală (cuvînt). Analiza lexicală se face considerînd întreg textul de definire drept context.

Introducerea textului de definire se poate efectua fie în modul *„batch"* (un fişier de tipul de reorganizare cerut de intrarea standard a sistemului) fie în *„conversaţional"* de la un terminal conversaţional.

Principial modul de funcţionare al analizorului lexical este următorul:

— la introducerea fiecărui simbol este apelată o procedură care citeşte în bufferul de intrare linii de 80 caractere ;

— fiecare linie citită este transferată într-un buffer propriu ;

— se lansează procedura principală a compilatorului care execută analiza lexicală a frazei citite. Această procedură izolează cuvintele din text detectînd separatorii de limbaj. Un cuvînt este definit ca un şir de caractere cuprins între doi separatori ai limbajului.

Compilatorul detectează unităţile sintactice şi în funcţie de cuvintele cheie rezervate stabileşte, pentru fiecare caracteristică, o structură internă.

Analiza sintactică a frazei se realizează conform regulilor de producţie asociate fiecărui tip de caracteristică.

Pentru construirea structurii interne convenim faptul că analizatorul LDD utilizează următoarele tabele de memorie :

— DICOBCK, pentru descrierea internă a caracteristicilor ;

— BROU, pentru extensii ale anumitor caracteristici ;

— DICONOM, pentru construirea unui dicţionar al identificatorilor.

Construirea structurii interne se face luînd în considerare fiecare caracteristică elementară şi legăturile existente între caracteristici, în mod strict paralel cu textul de definire.

Conţinutul dicţionarului

Pentru fiecare caracteristică elementară detectată în textul de definire se construiesc linii de intrare într-un tabel. Structura logică a liniilor din tabel poate fi, de exemplu, următoarea:

— adresa atribuită caracteristicii de către compilator. Stabilirea adresei se efectuează în conformitate cu poziţia secvenţială a caracteristicii în textul de definire. Deoarece, structura internă a bazei de date este manipulată prin mecanismul de paginare această adresă este tratată ca adresă virtuală a spaţiului virtual destinat structurii interne;

— identificatorul caracteristicii;

— punctatori spre celelalte caracteristici definite în structură care permit evidenţierea relaţiilor de incluziune structurală a unei caracteristici (caracteristica „tată", caracteristica „frate", caracteristica „fiu");

Conţinutul cîmpurilor destinate acestor pointeri pot avea diverse interpretări, funcţie de tipul caracteristicii;

— tipul intern al caracteristicii (de exemplu 2 — entitate, 3 — invers etc.);

— numărul maxim de realizări pentru entităţi;

— condiţiile de validare standard a valorilor atribuite;

— adresa virtuală de origine a caracteristicii în spaţiul virtual destinat datelor. Această adresă virtuală poate fi:

a) *absolută*, pentru caracteristicile definite la cel mai înalt nivel;

b) *relativă*, pentru caracteristicile definite la orice alt nivel.

— dimensiunea spaţiului de memorie rezervat caracteristicii;

— numărul intrării şi dimensiunea alocată în dicţionar pentru caracteristicile declarate chei de acces.

Pentru formarea unei îndemînări practice de construire a unei structuri interne vom prezenta ulterior interpretarea codului intern pentru SOCRATE V1.5.

Modificări ale LDD-SOCRATE în versiunea V 1.6.R-I.T.C.I.

În această versiune LDD a fost modificat pentru a permite:

— definirea caracteristicilor de tip ⟨*zecimal împachetat cu virgula*⟩;

— definirea poziţiei virgulei la caracteristicile de tip ⟨*zecimal împachetat*⟩ şi ⟨*zecimal despachetat*⟩ din formal;

— definirea caracteristicilor de tip ⟨*binar*⟩ în formal;

— definirea caracteristicilor de tip ⟨*formal pentru sortare*⟩;

— schimbarea modului de acces la dicţionar;

— definirea dinamică a cheilor de acces;

— stocarea tabelelor de indecşi asociate caracteristicilor de tip ⟨*referire cu inel*⟩ în spaţiul DICO (dicţionar) al bazei de date (în V.15 acestea sînt stocate în spaţiul FICH).

Caracteristica de tip ⟨zecimal⟩ (DECIMAL)

Rol : descrierea atributelor (cîmpurilor) de tip zecimal împachetat cu virgulă ;
Sintaxa :
⟨*identificator*⟩ DECIMAL [(] ⟨n_1⟩ [V⟨n_2⟩][)]][⟨*declarație acces*⟩]
unde :
⟨n_1⟩ : numărul maxim de cifre al părții întregi, $\langle n_1\rangle \in [0,15] \cap N$;
⟨n_2⟩ : numărul maxim de cifre al părții zecimale, $\langle n_2\rangle \in [0,7] \cap N$;

DILATE : definește un șir de caractere numerice reprezentate în zecimal împachetat.

Date acceptate : valori numerice reprezentate în zecimal împachetat din intervalul $[-n_1 \, V \, n_2, +n_1 V n_2] \cap R$;

Condiții de validare : sînt acceptate numai valorile care aparțin intervalului declarat ;

Spațiul ocupat (s) : se calculează cu formula :

$$(\langle n_1\rangle + \langle n_2\rangle)/2 = Q + R, \text{ dacă } R = \begin{cases} 0, \text{ atunci } s = Q \text{ octeți ;} \\ 1, \text{ atunci } s = Q + 1 \text{ octeți} \end{cases}$$

Structura și semnificația spațiului ocupat :

Fiecare caracter, din șirul rezervat, conține două cifre zecimale în afară de ultimul care conține o cifră și semnul în hexa (C sau F pentru numere pozitive și D pentru numere negative).

Cadraj : octet.

Operații admise : comparații și operanzi indirecți în calcule aritmetice.

Exemple :

CANTITATE 9 V3 — **9 intregi și 3 zecimale ;**
PREȚ-UNITAR 5 V2 — **5 intregi și 2 zecimale**

Modificarea caracteristicilor din ⟨formal⟩

1) Declararea caracteristicilor de tip ⟨*zecimal despachetat/împachetat*⟩ cu virgulă.
⟨*zecimal despachetat*⟩ : : = ⟨*identificator*⟩ DILATE [(][n_1]n_2[Vn_3][)]
⟨*zecimal împachetat*⟩ : : = ⟨*identificator*⟩ PACKE [(][n_1]n_2[V n_3][)]
unde :
⟨n_1⟩ : definește poziția de început a zonei de memorie adresată de ⟨*identificator*⟩. În lipsa declarării lui ⟨n_1⟩ începutul zonei adresate de ⟨*identificator*⟩ este dat de poziția (*pd*) în care s-a ajuns prin definirile anterioare +1. Variabila ⟨n_1⟩ servește, în esență, la redefinirea unor ⟨caracteristici formal⟩. Redefinirea nu are sens decît asupra unor caracteristici definite anterior (care au rezervat spațiu de memorie) deoarece ea nu rezervă spațiu de memorie ci oferă o altă viziune asupra zonelor definite. În această accepțiune valorile lui ⟨n_1⟩, ⟨n_2⟩ și ⟨n_3⟩ trebuie să satisfacă relația :
– pentru DILATE : $(n_1 + n_2 + n_3 - 1) \leqslant pd$;
– pentru PACKE : $[n_1 + (n_2 + n_3 + 2)/2] - 1 \leqslant pd$;

$\langle n_2 \rangle$: numărul maxim de cifre pentru partea întreagă ;
$\langle n_3 \rangle$: numărul maxim de cifre pentru partea zecimală ;

$$\langle n_2 \rangle \in [0,15] \cap N \; ; \; \langle n_3 \rangle \in [0,7] \cap N \; ;$$

DILATE : zona de memorie, descrisă, va fi tratată ca un șir de caractere, de lungime $(n_2 + n_3)$ octeți, reprezentate în zecimal despachetat ;

PACKE : zona de memorie descrisă va fi tratată ca un șir de caractere, de lungime $(n_2 + n_3 + 2)/2$ octeți, reprezentate în zecimal împachetat.

Exemple :

```
FORMAL VIRGULĂ
 DEBUT
  PRET DILATE (5 V2)          /* 7 octeți */
  CANTITATE PACKE 9 V3        /* 7 octeți */
  IDENTN DILATE (1 10)        /* 10 octeți */
  IDENTA MOT    (1 10)        /* 10 octeți */
  IDENTV DILATE (3 8 V4)      /* 12 octeți */
FIN
```

2) Declararea caracteristicilor de tip ⟨*binar*⟩
⟨*identificator*⟩ BINAIRE [(][$\langle n_1 \rangle$]$\langle n_2 \rangle$[)]
unde :
$\langle n_1 \rangle$: definește poziția de început a zonei de memorie adresată de ⟨*identificator*⟩ ;
$\langle n_2 \rangle$: :=2/4
— 2 : caracteristici binare reprezentate pe semicuvînt ;
— 4 : caracteristici binare reprezentate pe cuvînt.

BINAIRE : definește o zonă de memorie al cărei conținut va fi tratat ca număr binar. Pentru a simplifica algoritmii de tratare a acestor zone (și munca de programare) realizatorii modificării au introdus restricția ca adresa la care sînt definite în buffer aceste caracteristici să fie multiplu de 2 (pentru $\langle n_2 \rangle = 2$) sau de 4 (pentru $\langle n_2 \rangle = 4$).

Observație :

Utilizarea acestui tip de caracteristici mărește puterea de reprezentare a tipului de date externe acceptate de sistem (duce la o economisire de timp de execuție U.C. și I/0 prin eliminarea conversiilor și, respectiv, micșorarea numărului de octeți manipulați). Pe de altă parte, asocierea acestei declarații cu instrucțiunea FORMAT, permite o utilizare eficientă a acesteia prin eliminarea unor artificii pe care utilizatorul este obligat să le facă în SOCRATE V.1.5 pentru a prelucra, conform necesităților sale, buffere structurate în „format de reprezentare internă" a realizărilor de entitate.

3) Utilizarea redefinirii la declararea unui ⟨*formal logic repetitiv*⟩

Rol : specificarea poziției de început a redefinirii în cadrul unui ⟨*formal logic*⟩, cu un ⟨*formal logic repetitiv*⟩.

Sintaxa :

FORMAL[(][$\langle n_1 \rangle$]$\langle n_2 \rangle$[)]⟨*identificator*⟩
 DEBUT
 ⟨*lista de caracteristici*⟩
 FIN

$\langle n_1 \rangle$: poziția, din ⟨*formalul logic*⟩ de care aparține, de unde începe redefinirea ;
$\langle n_2 \rangle$: numărul maxim de realizări ale ⟨*formalului logic repetitiv*⟩.

Caracteristici de tip ⟨formal pentru sortare⟩ (FORMAL SORT)

Rol : descrierea articolului de sortare.

Aceste caracteristici reprezintă caracteristici „fictive", adică, pentru ele, nu se rezervă spațiu apriori în baza de date nici în memoria centrală. Aceste caracteristici permit structurarea unui buffer de intrare/ieșire, desemnat de utilizator la nivelul limbajului de manipulare, conform structurii unor articole care fac obiectul unei operații de sortare realizată conform definiției asociate acestora.

Sintaxa :

FORMAL SORT ⟨*identificator*⟩ AVEC CLE {UNIQUE/DOUBLE/ADDITION}
DEBUT
  CLE
    DEBUT
      ⟨*caracteristici cheie*⟩
  FIN
      ⟨*caracteristici articol*⟩
FIN

⟨*caracteristici cheie*⟩ : : = ⟨*cuvînt formal*⟩/⟨*zecimal despachetat*⟩/
  ⟨*cuvînt formal*⟩⟨*caracteristici cheie*⟩/
  ⟨*zecimal despachetat*⟩⟨*caracteristici cheie*⟩
  ⟨*caracteristici articol*⟩ : : = ⟨*caracteristică articol*⟩/
  ⟨*caracteristică articol*⟩⟨*caracteristici articol*⟩

⟨*caracteristică articol*⟩ : în funcție de tipul de cheie declarat avem :

a) CLE ADDITION :
⟨*caracteristică articol*⟩ : : = ⟨*cuvînt formal*⟩/⟨*binar*⟩/⟨*zecimal împachetat*⟩

b) CLE UNIQUE/DOUBLE} :
⟨*caracteristică articol*⟩ : : = ⟨*cuvînt formal*⟩/⟨*binar*⟩/
⟨*zecimal împachetat*⟩/⟨*zecimal despachetat*⟩

Specificații de utilizare

Definirea unui formal de sortare se poate realiza sub procesorul :

– D (Definition) : la definirea structurii bazei de date sau prin adăugare la structură ;
– R (Requette) : la compilarea – execuția programelor de interogare ;
– M (Macrogenerateur) : modificat pentru a accepta blocul : DEFORM pentru definirea caracteristicilor de tip ⟨formal⟩.

Pentru o bază de date oarecare formatele logice de sortare se definesc numai la nivelul exterior structurii acesteia.

Nu este admisă redefinirea caracteristicilor din acest tip de formal : caracteristicile din blocul CLE sînt considerate părți componente ale cheii iar celelalte caracteristici părți componente ale corpului articolului de sortare.

În funcție de modul de selecție a formei cheii formalul de sortare poate fi :

– cu cheie unică (AVEC CLE UNIQUE) : dacă există articole duplicate (cu aceiași cheie) atunci va fi luat în considerare primul articol introdus, celelalte (duplicatele) fiind omise ;

– cu cheie dublă (AVEC CLE DOUBLE) : vor fi sortate toate articolele indiferent dacă sînt duplicate sau nu ;

– cu cheie cu adunare (AVEC CLE ADDITION) : la toate articolele cu aceiași cheie se va executa operația de adunare automată a tuturor cîmpurilor (BINAIRE sau PACKE) care au fost modificate și aparțin corpului articol (articolele duplicate –

din punct de vedere al cheii şi conţinutului corpului articolului se iau în considerare). Pentru caracteristica de tip MOT se va muta prima valoare definită întîlnită. Blocul CLE este obligatoriu şi trebuie să aibă o lungime mai mare ca 0 (zero).

Exemplu :

Presupunem că avem la intrare veniturile realizate de persoanele unei întreprinderi în ultimele 12 luni şi dorim, prin sortare, să obţinem venitul total realizat.

Formalul necesar acestei operaţii este :

```
FORMAL SORT AVEC CLE ADDITION
 DEBUT
   CLE /* definire cheie de sortare */
     DEBUT
       MARCA DILATE 13 /* SAALLZZ NNNC */
   FIN /* sfîrşit definire cheie de sortare */
   /* definire corp articol */
  NUME MOT 16
  PRENUME MOT 16
  RETRIBUTIA PACKE 15 V2
 FIN /* sfîrşit definire corp articol şi formal */
```

Indiferent de numărul de articole/persoana (MARCA) vom avea o singură realizare pe valoare diferită a „mărcii" care conţine în cîmpul :

— NUME — prima valoare întîlnită ;

— PRENUME — prima valoare întîlnită

— RETRIBUŢIA : suma tuturor valorilor atribuite acestui cîmp pentru toate articolele citite.

Remarca : mărimea bufferului ⟨*ds*⟩ necesar unei caracteristici de tip ⟨*formal pentru sortare*⟩ (care trebuie rezervat de utilizator cu comanda. OPTION, parametrul DS) se calculează, în funcţie de tipul cheii, astfel :

— *CLE* {UNIQUE/DOUBLE} :

⟨*ds*⟩ :=((LART + L CHEIE) ÷ 2 + L CHEIE) ÷ 4 + 24, octeţi ;

— *CLE ADDITION* :

⟨*ds*⟩ : = ((LART + L CHEIE) ÷ 2 + 2 ∗ NC + L CHEIE) ÷ 4 + 24, octeţi ;

unde :

LART : lungime în octeţi a corpului articolului ;

LCHEIE : lungime în octeţi a corpului cheii ;

NC : numărul de cîmpuri definite în corpul articolului ;

÷ i : valoarea va fi adusă la multiplu superior de i (rotunjire prin adaos). Operaţia „calculează multiplu de i (M)" este definită astfel :

$$(n) \div i :: = n/i = Q + R, \text{ dacă } R = \begin{cases} 0, \text{ atunci } M = n \,; \\ \neq 0, \text{ atunci } M = n + (i - R) \,; \end{cases}$$

Pentru exemplul anterior avem :

L CHEIE : = 13 octeţi ;

L ART : = 16 + 16 + (15 + 2 + 2)/2 : = 32 + 9 : = 41 octeţi ;

2∗NC : = 2∗1 : 2 ;

$$\langle ds \rangle \quad : = ((41 + 13) \div 2 + 2 + 13) \div 4 + 24 : =$$
$$(54 \quad + 2 + 13) \div 4 + 24 : =$$
$$(69) \div 4 \quad + 24 : =$$
$$72 \quad + 24 : = 96 \text{ octeți.}$$

Schimbarea modului de acces la dicționar

Această schimbare a fost efectuată pentru a permite sortarea directă a datelor din baza de date.

Sintaxa de definire a caracteristicilor cu cheie (⟨*declarație acces*⟩) a fost modificată astfel:
{⟨*cuvînt*⟩/⟨*valoare numerică*⟩/⟨*zecimal*⟩} AVEC CLE
[{UNIQUE/DOUBLE}][{AVANT/ARRIERE}][CHAINE{SIMPLE/DOUBLE}]
[[NON]ORDONNE][(⟨*factor umplere*⟩)] FIN

În această declarație numai cuvintele subliniate sînt active celelalte sînt tratate drept comentarii și au fost reținute pentru a păstra compatibilitatea declarației cu ⟨*declarație acces*⟩ din V.1.5. Valorile atribuite caracteristicii sînt clasate în dicționar fără a mai utiliza funcția de „hash-cod"
UNIQUE : valorile atribuite caracteristicii sînt unice (cheie discriminată) ;
DOUBLE : sînt admise valori identice (cheie rapidă).

Valorile clasate în dicționarul asociat caracteristicii, prin ⟨*declarație acces*⟩, sînt parcurse, la interogare (utilizarea opțiunii PAR sau AVEC), în ordine :
– AVANT : crescătoare (implicit) ;
– ARRIERE : descrescătoare.
⟨*factor umplere*⟩ : factor de umplere a subpaginilor dicționarului asociat caracteristicii ⟨*factor umplere*⟩ $\in [0,50] \cap N$ și reprezintă procentul de neocupare a paginilor de dicționar (cît se lasă liber pentru eventuale adăugiri). Valoarea implicită atribuită este de 50.

La încărcarea inițială a bazei de date, dacă datele sînt sortate după caracteristicile declarate chei de acces, este indicată declararea factorului de umplere pentru a asigura o încărcare optimă a paginilor de dicționar. *După încărcarea inițială a bazei de date se va elimina declarația ⟨factor umplere⟩ prin recompilarea structurii* (utilizarea procesorului LDD sub controlul nivelului ACTI TB : 49).

Definirea dinamică a cheilor de acces

Modul de organizare și gestiune a direcționarului a permis definirea/suprimarea uneia sau mai multor ⟨*declarații acces*⟩ la o bază de date încărcată fără să afecteze major structura fizică a acesteia.

În SOCRATE V.1.5 definirea/suprimarea unei ⟨declarații acces⟩ presupunerea, parcurgerea următorilor pași :
1 : salvarea datelor stocate în baza de date sub un format stabilit de utilizator ;
2 : formatarea spațiilor DICO și FICH ;
3 : compilarea structurii bazei de date ;
4 : încărcarea bazei de date utilizînd datele salvate la 1 : și eventual alte date.

În V.1.6.R sînt eliminate aceste operații deoarece alocarea dicționarelor (spațiul dicționar, în ambele versiuni, este o resursă partajată în principal pentru caracteristicile declarate chei de acces) nu se mai face strict paralel cu textul de definire (în ordinea secvențială a apariției caracteristicilor declarate chei de acces) ci strict paralel cu ordinea declarării temporale (indiferent de locul ocupat în structură).

Operațiile pe care trebuie să le execute utilizatorul depind de modificarea efectuată asupra structurii astfel:

a) definirea unei ⟨*declarații acces*⟩ la o caracteristică care nu este cheie:

a1 : — se modifică definiția caracteristicii adăugînd declarația de acces dorită de utilizator;

a2 : — se recompilează structura bazei de date sub forma:

% ACTI TB : 49/* comanda de activare-adăugare/redefinire structură */
% RUN FN : D/* comanda de apel procesor (compilator LDD) */
DEBUT /* noua structură va lua locul celei vechi */
⟨*structura*⟩
FIN
?

Dacă noua structură este corectă atunci se editează mesajul:

ON A CREE NOUVEAU DICTIONAIRE POUR LES VARIABILES
(s-a creat un dicționar nou pentru variabilele)

$\langle identificator \rangle_1$ $\langle n_1 \rangle$
.
.
.
$\langle identificator \rangle_i$ $\langle n_i \rangle$
.
.
.

unde:

$\langle identificator_i \rangle$: identificatorul caracteristicii i declarată cheie de acces;
$\langle n_i \rangle$: adresa de început a dicționarului asociat caracteristicii $\langle identificator \rangle_i$.

a3 : — se activează dicționarul asociat fiecărei caracteristici declarate chei de acces (cele pentru care sistemul a dat mesaj de creare a unui nou dicționar) prin cereri efectuate în LMD sub forma:

% RUN FN : R /* apel procesor LMD în „batch processing" */
POUR TOUT ⟨nume entitate$_i$⟩ X_i
M ⟨identificator⟩$_i$; ⟨identificator⟩$_i$ DE X_i
FIN ?

unde:

$\langle nume\ entitate_i \rangle$: identificatorul caracteristicii de tip entitate care conține caracteristica declarată cheie cu numele $\langle identificator_i \rangle$. Evident dacă $\langle nume\ entitate_i \rangle$ este o entitate imbricată de un nivel inferior atunci utilizatorul trebuie să completeze citația (citațiile) necesară(e) calificării și cuantificării totale a acestui nivel (pentru lămuriri suplimentare vezi capitolul 4).

Activarea dicționarului reprezintă operația de creare a tebelei de indecși asociată caracteristicii ⟨*cheie de acces*⟩, tabelă care conține o intrare asociată fiecărei valori atribuite cheii.

b) suprimarea unei ⟨*declarații acces*⟩ la o caracteristică ⟨*cheie de acces*⟩ :

b1 : – se salvează, printr-un program utilizator, valorile asociate caracteristicii ;

b2 : – caracteristica se pune la nedefinit, prin cereri exprimate în LMD, pentru a distruge dicționarul. Dacă utilizatorul nu distruge dicționarul atunci spațiul ocupat de acesta devine indisponibil.
Cererile de punere la nedefinit pot fi de forma :

```
%, RUN FN : R /* apel procesor LMD */
POUR TOUT ⟨nume entitate⟩i
   M ⟨identificator⟩i = U
FIN ? (vezi a3:) ;
```

b3 : – se modifică definiția caracteristicii (suprimînd ⟨*declarație acces*⟩ ;

b4 : – se recompilează structura sub controlul lui ACTI TB : 49 ;

b5 : – se încarcă, în caracteristică, prin program utilizator valorile salvate la b1 :.

Din această procedură de lucru sînt obligatorii numai pașii b3 : și b4 :. Evident omiterea oricăruia din ceilalți pași are consecințe specifice fiecăruia și numai administratorul bazei de date este cel care decide dacă omisiunea este oportună sau nu.

Observații : eliminarea declarației de acces la o caracteristică ⟨*cheie de acces*⟩ presupune corectarea codului programelor care citează această caracteristică în cereri, sub controlul lui AVEC sau PAR, în sensul eliminării lui PAR și înlocuirii lui AVEC cu AYANT (modificările pot fi făcute cu ajutorul „editorului de texte" al SGBD-SOCRATE).

Dacă codurile de program modificate sînt sub formă de macroinstrucțiune sau program precompilat atunci este necesară recatalogarea acestora.

Exemplu complex de definire a structurii unei baze de date

Pentru a exemplifica modul de utilizare al LDD la descrierea structurii conceptuale am ales domeniul de activitate PERSONAL. Acest domeniu a fost ales din considerentul că cititorul este familiarizat cu datele sale personale și, în acest context, nu mai este necesară o prezentare în detaliu a acestei activități. Vom insista mai mult asupra anumitor relații descrise în schema conceptuală.
Atributele entităților vor fi prezentate în descriere si vor fi comentate, dacă este cazul. Această structură încorporează în general exemplele prezentate anterior în această lucrare.

În această descriere nu vom trata întregul domeniu de activitate PERSONAL și nici legăturile sale cu alte domenii. Descrierea are un rol didactic-aplicativ pentru a evidenția, pe de o parte, posibilitățile LDD iar, pe de altă parte, pentru a oferi un material de referință la exemplificările efectuate în capitolele care urmează.

Vom denumi baza de date care are această structură conceptuală BDDPERS. *Ea este reprezentată în DISPLAY-ul 1.* Descrierea structurii conceptuale cu ajutorul LDD va fi stocată în fișierul DEFSTR.

Cu aceste precizări structura conceptuală descrisă cu ajutorul diagramelor Bachman se prezintă ca în figura 3.4. Obținerea acestei scheme a fost dictată de genul de întrebări la care trebuie să răspundă baza de date. Printre aceste întrebări se regăsesc și următoarele :

– care este structura personalului pe sexe ?

– care este structura personalului pe profesii și ani de naștere ?

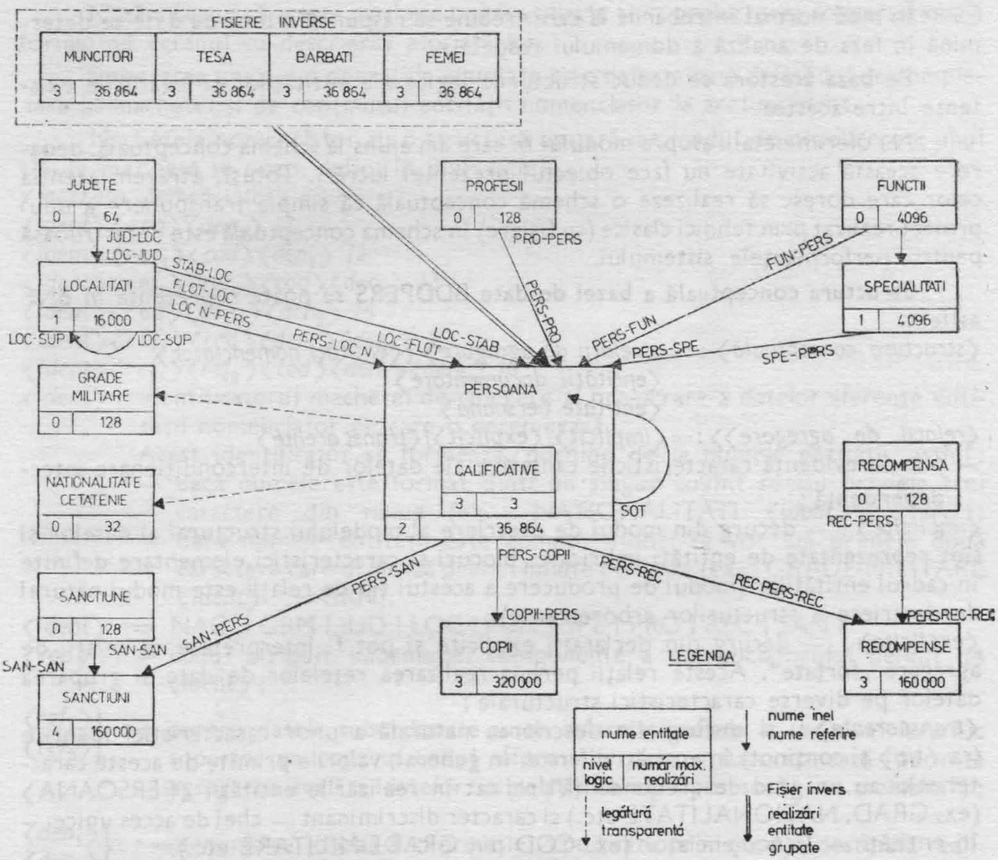


Fig. 3.4. Schema conceptuală a bazei de date BDDPERS (Bachman)

— care este structura personalului pe funcții și specialități?

— ce recompense a primit o persoană etc.

În această schemă entitățile GRADE-MILITARE și NAȚIONALITATE-CETĂȚENIE sînt descrise ca structuri punctuale.

Vor fi folosite numai ca dicționar pentru validare/afișare a informației respective. Elementul de acces la realizări va fi reprezentat de numărul de realizare pentru NAȚIONALITATE-CETĂȚENIE și de cheia unică (atributul COD) pentru GRADE-MILITARE.

Legătura SOȚ reprezintă o referire simplă pe nume de entitate (PERSOANA).

Între entitatea RECOMPENSĂ și SANCȚIUNE și entitatea PERSOANA avem o relație de tip „m ↔ n" materializată prin intermediul entităților de legătură RECOMPENSE și respectiv SANCȚIUNI. Aceste legături permit să răspundem la întrebări de genul:

— care sînt recompensele/sancțiunile primite de o persoană pe parcursul activității sale;

— care sînt persoanele care au primit aceeași recompensă/sancțiune.

În mod normal întrebările la care trebuie să răspundă o bază de date se determină în faza de analiză a domeniului respectiv.

Pe baza acestora se deduc structurile logice ale entităţilor şi legăturile existente între acestea.

Nu oferim detalii asupra modului în care am ajuns la schema conceptuală, deoarece această activitate nu face obiectul prezentei lucrări. Totuşi, atragem atenţia celor care doresc să realizeze o schemă conceptuală că simpla transpunere a unui proiect realizat prin tehnici clasice (cu fişiere) în schema conceptuală este dezastruoasă pentru performanţele sistemului.

Structura conceptuală a bazei de date **BDDPERS** se poate reprezenta în BNF astfel:

⟨*structura conceptuală*⟩ : : = ⟨*relaţii de agregare*⟩ (⟨*entităţi nomenclator*⟩
⟨*entităţii documentare*⟩
⟨*entitate persoana*⟩

⟨*relaţii de agregare*⟩⟩ : = ⟨*implicit*⟩/⟨*explicit*⟩/⟨*transparente*⟩

– pun în evidenţă caracteristicile calitative ale datelor de intercondiţionare-interdependenţă ;

⟨*implicite*⟩ : – decurg din modul de descriere al modelului structural al datelor şi sînt reprezentate de entităţi imbricate, blocuri şi caracteristici elementare definite în cadrul entităţilor (modul de producere a acestui tip de relaţii este modul natural de descriere a structurilor arborescente);

⟨*explicite*⟩ : – decurg din declaraţii explicite şi pot fi interpretate ca relaţii de agregare „forţate". Aceste relaţii permit realizarea reţelelor de date şi gruparea datelor pe diverse caracteristici structurale ;

⟨*transparente*⟩ : – decurg din descrierea naturală a unor caracteristici identice (ca ⟨*tip*⟩ şi conţinut) în entităţi diferite. În general, valorile primite de aceste caracteristici au un grad de „redondanţă" ridicat în realizările entităţii ⟨PERSOANA⟩ (ex. GRAD, NAŢIONALITATE etc.) şi caracter discriminant – chei de acces unice – în entităţi – la nomenclator (ex. COD din GRADE-MILITARE etc.).

⟨*entităţi-nomenclator*⟩ : reprezintă entităţi „externe", „personal", şi cuprind informaţiile în clar aferente datelor codificate care caracterizează o „persoană" şi eventual grupări ale „persoanelor" care prezintă aceeaşi valoare pentru o anumită caracteristică (ex. : funcţie, profesie etc.).

⟨*entităţi documentare*⟩ : – reprezintă entităţi care conţin datele cu anumit grad de repetabilitate temporală aferente unei „persoane" (ex. : activităţi-desfăşurate). Realizările acestui tip de entităţi sînt grupate, în număr variabil, şi ataşate unei anumite „persoane". Datele conţinute de acest tip de entitate (corespunzătoare caracteristicilor definite în entitate) pot avea legături cu entităţile nomenclator (prin relaţii de agregare explicite şi/sau transparente) ;

⟨*entitatea persoana*⟩ : – conţine datele proprii unei persoane (angajat) la un moment dat, iar prin relaţiile de agregare stabilite la nivel fizic (prin declaraţiile structurale) şi la nivel logic (prin combinaţii realizate în programele de exploatare) cu entităţi nomenclator care au aceleaşi valori pentru anumite caracteristici (ex. : aceeaşi funcţie, profesie, specialitate etc.) şi respectiv informaţii istorice asupra modului de evoluţie a unor date ataşate unui individ (ex. : distincţii, activităţi-desfăşurate etc.).

Pentru prelucrarea în „batch-processing" fiecărei entităţi nomenclator i s-a ataşat o machetă de culegere şi prelucrare a datelor, machetă utilizată pentru : creare, adăugare, modificare, ştergere.

Notăm faptul că aceste machete pot fi utilizate și în prelucrarea conversațională formatînd ecranul cu descrierea asociată lor.

Selectarea uneia din operațiile enunțate este determinată de modul de completare al machetei și de conținutul entității nomenclator la acel moment.

Machetele nomenclator au o structură unitară iar modul de producere a unui tip de machetă se poate defini în BNF astfel:

⟨*macheta nomenclator*⟩ : :=
⟨*ident*⟩⟨*cod*⟩$\langle den_1 \rangle$ /1
⟨*ident*⟩$\langle ref_1 \rangle$⟨*cod*⟩$\langle den_1 \rangle$ /2
⟨*ident*⟩$\langle ref_1 \rangle\langle ref_2 \rangle$⟨*cod*⟩$\langle den_1 \rangle$ /3
⟨*ident*⟩⟨*cod*⟩$\langle den_1 \rangle\langle den_2 \rangle$ /4
⟨*ident*⟩$\langle ref_1 \rangle$⟨*cod*⟩$\langle den_1 \rangle\langle den_2 \rangle$ /5
⟨*ident*⟩$\langle ref_1 \rangle\langle ref_2 \rangle$⟨*cod*⟩$\langle den_1 \rangle\langle den_2 \rangle$ /6

⟨*ident*⟩ : identificatorul machetei de culegere și prelucrare a datelor aferente entității nomenclator pe care o desemnează.
Acest identificator se formează, pornind de la numele entității, astfel:
- dacă numele este format dintr-un singur cuvînt se iau primele trei caractere din nume (ex.: NATIONALITATI ⟨*ident*⟩ : :=NAT);
- dacă numele entității este compus atunci se vor lua primele două caractere care urmează primei liniuțe de unire (ex.: GRADE-MILITARE ⟨*ident*⟩ : :=GRM).

⟨*ident*⟩: := NAC I GRM I JUD I LOC I FUN I SPE I PRO I REC I SAN I

⟨*cod*⟩ : : = codul atribuit informației componente a nomenclatorului definit de ⟨*ident*⟩;

$\langle ref_1 \rangle$, $\langle ref_2 \rangle$: := pentru datele subordonate unor informații aflate la un nivel ierarhic superior va conține codul informației care o subordonează (permite agregarea reală a datelor în relații de tip „posesor" — „membru");

$\langle den_1 \rangle$, $\langle den_2 \rangle$: : =denumirea în clar a informației, codificate prin ⟨cod⟩, destinată realizării entității desemnate de identificatorul ⟨*ident*⟩.

Fiecare machetă este tratată de un program care realizează operațiile de creare-actualizare-modificare, identificat prin CAM — ⟨*ident*⟩ al cărui principiu de funcționare este următorul:

CAM1. anulare variabile;
CAM2. declanșare ciclu de prelucrare;
CAM3. citește o cartelă;
CAM4. DACĂ cîmpul ⟨*ident*⟩ = '$$$' ATUNCI ieșire din ciclu (CAM8).
CAM5. DACĂ există o realizare cu codul ⟨cod⟩
    ATUNCI
        DACĂ cîmpul $\langle den_1 \rangle$ este vid
            ATUNCI realizarea va fi ștearsă;
            ALTFEL se modifică denumirea realizării prin conținutul lui $\langle den_1 \rangle$ și eventual $\langle den_2 \rangle$;
    ALTFEL
        DACĂ cîmpul $\langle den_1 \rangle$ este vid
        ATUNCI se editează un mesaj de eroare non fatal
        ALTFEL se crează o realizare cu codul ⟨cod⟩ și denumirea $\langle den_1 \rangle$ și eventual $\langle den_2 \rangle$;

CAM6. **DACĂ** cîmpul $\langle ref_1 \rangle (\langle ref_2 \rangle)$ este prezent
ATUNCI DACĂ există o realizare a entității referente
ATUNCI se realizează legătura ;
ALTFEL se editează un mesaj non-fatal
CAM7. reluare ciclu (CAM3) ;
CAM8. editare mesaj de sfîrșit.

În figura 3.5 oferim structurile machetelor asociate nomenclatoarelor pentru o prelucrare de tip „batch processing“ sau conversațională cu ecran formatat. Macheta este formatată pe linii de maxim 80 caractere. Zonele denumite „spațiu“ nu sînt necesare, de aceea nu vor fi analizate și tratate de programul care prelucrează macheta respectivă.

Ansamblului machetelor definite le corespunde o descriere de tip ⟨*formal*⟩ denumită MACHETA stocată în fișierul MACNOM. De asemenea, pentru încărcarea datelor asociate unei persoane am definit două machete descrise în formalul PERS.

Structura acestor machete poate fi dedusă, ca exercițiu, din această descriere. Această caracteristică de tip ⟨*formal*⟩ are descrierea stocată în fișierul MACPER.

În continuare vom prezenta modul în care utilizatorul poate compila textele definite în acest paragraf. Prezentarea are rolul de a permite cititorului să facă exerciții de definire pe implementarea de care dispune. Mai mult dacă se studiază LMD atunci avînd o structură definită pot fi scrise programe în LMD pentru testarea diferitelor comenzi ale acestuia.

NAC = NAționalitate - Cetățenie

ident	cod	den 1	spațiu
NAC	X	X(25)	X(51)

GRM::= GRade - Militare

ident	cod	den 1	spațiu
GRM	999	X(27)	X(47)

JUD::= JUDețe

ident	cod	den 1	spațiu
JUD	99	X(15)	X(60)

LOC::= LOCalități

ident	ref 1 (JUD)	ref 2 (LOC)	cod	den1	den 2	spațiu
LOC	99	9(6)	9(6)	X(30)	X(20)	X(13)

FUN::= FUNcții

ident	cod	den 1	den2	spațiu
FUN	9(4)	X(30)	X(20)	X(23)

SPE::= SPEcialități

ident	ref 1 (FUN)	cod	den 1	den 2	spațiu
SPE	9(4)	9(4)	X(30)	X(30)	X(9)

PRO::= PROfesii

ident	cod	den 1	den2	spațiu
PRO	99	X(30)	X(30)	X(15)

REC::= RECompense

ident	cod	den 1	den 2	spațiu
REC	9(3)	X(30)	X(30)	X(9)

SAN::= SANcțiune

iden	cod	den1	den2	spațiu
SAN	9(3)	X(30)	X(30)	X(14)

Fig. 3.5. Structura formatelor articolelor pentru încărcarea – actualizarea entităților de tip-NOMENCLATOR.

STRUCTURA CONCEPTUALA A BAZEI DE DATE

```
debut
/* variabile de lucru                                    */
raspuns mot
  rasp  mot 15
continuati (4 2) (da nu y n )
mnemonica mot 4
codnum    de 1 a 999999999
codalf     mot
caracter (64 1) (1 2 3 4 5 6 7 8 9 a b c d e f  )
/*------------------------------------------------------*/
/*                                                      */
muncitori      inverse tout persoana
tesa           inverse tout persoana
invgen         inverse tout persoana
barbati        inverse tout persoana
femei          inverse tout persoana.
/*------------------------------------------------------*/
/*                                                      */
/*  entitatea : <persoana>                              */
/*          <<         >>                               */
/* -----------<< nucleu >>    --------------------------*/
entite 36864   persoana
 debut
  /* <cod>::= saalljjnnnc                               */
  /* -s: sex/secol;                                     */
  /* -aallzz: data nasterii;                            */
  /* -jj: judetul; -nnn: numar de ordine;               */
  /* -c: cifra de control.                              */
  cod          mot 13 avec cle unique fin
  nume         mot 16 avec cle ordonne fin
  prenume      mot 16
  data-nasterii
   debut
    an   de 1900 a 2100
    luna de 1 a 12
    zi   de 1 a 31
    pers-locn refere locn-pers de un localitati
  fin
  cetatenie       (15 1) (1 2 3 4 5 6 7 8 9 a b c d e f )
  nationalitate   (15 1) (1 2 3 4 5 6 7 8 9 a b c d e f )
  stare-civila    (4  1) (1 2 3 4 )
  apartenenta     (4 5)  (pcr utc odus nm)
  sanct-pol       (15 3) (ep vb vba av mus ms )
  data-pcr
   debut
    an   de 1921 a 2100
    luna de 1 a 12
    zi   de 1 a 31
  fin
  situatia-militara
   debut
    livret
     debut
      seria mot 3
      nr   de  0  a 999999
    fin
    specialitate  de 0  a 999
    comisariat    de 0  a 99
    /* comisariat := cod judet ; */
    grad          de 0  a 99
    obligarii   (7  1) ( 1 2 3 4 5 6 7 )
   fin
   grupa-sanguina (4  4) (oi aii biii abiv)
   domiciliu
```

```
debut
 stabil
  debut
   loc-stab refere stab-loc de un localitati
   str      mot 26
   nr       mot 4
   bl       mot 3
   sc       mot 2
   etaj     de  0 a 32 (64)
   ap       de  0 a 999
   mediu    (2  1) (1 2)
   naveta
    debut
     distanta de 0 a 99
     an-apr   de 1900  a  2100
     luna-apr de 1 a 12
     ziua-apr de 1 a 31
   fin
   locuinta
    debut
     supr-loc      de 1 a 999
     nr-camere     de 1 a 32
     nr-persoane de 1 a 99
     proprietate (7  1) (1 2 3 4 5 6 7 )
   fin
   buletin-ident
    debut
     seria       mot 2
     /* compilatorul <ldd> accepta si linii vide */
     nr          de 0 a 999999
     an-emitere  de 1950 a 2100
    fin
  fin
  flotant
   debut
   loc-flot refere flot-loc de un localitati
   str      mot 26
   nr       mot 4
   bl       mot 3
   sc       mot 2
   etaj     de 0 a 32 (64)
   ap       de 0 a 99
   mediu    (2  1) (1 2 )
      /* 1: urban; 2: rural. */
  fin
 fin
 /* copii persoanei                  */
 pers-copii     anneau
 sot            refere un persoana
 pers-fun       refere fun-pers de un functii
 pers-pro       refere pro-pers de un profesii
 pers-spe       refere spe-pers de un specialitati
 pers-san anneau
 pers-rec       anneau
      avec chaine double fin
 entite 3 calificative
  debut
  an        de 1977 a 2100
  calific (4  2) (fb b ns st)
                       /* fb: foarte bun; */
                       /* b : bun;        */
                       /* ns: nesatisfacator; */
                       /* st: satisfacator.   */
 fin
fin
/*  entitatea : <judete>               */
entite   (64)     judete
```

```
    debut
      jud-loc  anneau
      cod      de 1 a 99  avec cle unique fin
      den1     mot
    fin
    /*   entitatea : <localitai>                      */
    entite  16000 localitati
     debut
      loc-jud      refere  jud-loc de un judete
      /* localitate de care apartine ierarhic */
      sup-loc      refere  loc-sup de un localitati
      /* localitati subordonate administrativ  */
      loc-sup      anneau
      /* persoanele nascute in aceiasi localitate */
      locn-pers    anneau
      /* persoanele cu domiciliul stabil in localitate */
      stab-loc     anneau
      /* persoanele cu domiciliul flotant in localitate */
      flot-loc     anneau
      cod          de 1 a 999999 avec cle unique fin
      den1         mot
      den2         mot ( 10 )
    fin
    /* entitatea : <grade-militare>               */
    entite 128 grade-militare
     debut
      cod   de 0 a 99 avec cle unique fin
      den1  mot
    fin
    /* entitatea : <nationalitate-cetatenie>            */
    entite ( 32 )  nationalitate-cetatenie
     debut
      /* cheia realizarii este <numar de realizare>        */
      den1   mot
    fin
    /* entitatea : <functii>                              */
    entite   4096  functii
     debut
      fun-spe   anneau
      fun-pers  anneau
      cod       de 1 a 9999999 avec cle unique fin
      den1      mot
      den2      mot
    fin
    /* entitatea : <specialitati>                     */
    entite   4096  specialitati
     debut
      spe-fun refere fun-spe de un functii
      spe-pers  anneau
      cod       de 1 a 9999999 avec cle unique fin
      den1      mot
      den2      mot
    fin
    entite  320000  copii
     debut
      copii-pers refere pers-copii de un persoana
      nr-copil          de 1 a 32
      nume              mot 16
      prenume           mot 14
      loc-nastere       de 1 a 999999
      data-nasterii
       debut
        an   de 1916 a 2100
        zi   de 1 a 31
        luna de 1 a 32
     fin
      rang-alocatie de 0 a 99
```

```
      invaliditate  de 0 a 9
      sistare-aloc  de 0 a 9
      informatii    texte  (4  60)
     fin
    /* ---------------------------------------------------*/
   /*                  <<atentie>>                      */
   /*   urmeaza doua descrieri de relatii ''m <--> n''   */
   /* --------------- ----- ------------------------------*/
    /* entitatea  : <sanctiune>                           */
    entite  128 sanctiune
     debut
      sanct anneau
      cod   de 1 a 99 avec cle unique fin
      den1  mot
      den2  mot
    fin
    /* entitatea : <sanctiuni>                            */
    entite 160000 sanctiuni
     debut
      san-sanct  refere sanct de un sanctiune
      san-pers   refere pers-san de un persoana
      nr-san     de 0 a 99
      procent    de 0 a 99
      cl-cat     de 0 a 99
      gr-tr      de 0 a 99
      retributie de 0 a 29999
      functie    de 1 a 4096
         /* atentie ] este numarul de realizare           */
         /*---------------------------------              */
      an        de 1944 a 2100
      luna      de 1 a 12
      zi        de 1 a 31
      motiv     texte (255  60)
    fin
    /* entitatea : <profesii>                             */
    entite  128 profesii
     debut
     pro-pers  anneau
      cod       de 0 a 99 avec cle unique fin
      den1      mot
    fin
    /* entitatea : <recompensa>                           */
    entite  128  recompensa
     debut
      rec-pers anneau avec chaine double fin
      cod      de 1 a 128 avec cle unique fin
      den1     mot
      den2     mot
    fin
    /* entitatea : <recompense>                           */
   entite  160000 recompense

     debut
      perrec-rec refere rec-pers de un recompensa
                                 avec chaine double fin
      recper-rec refere pers-rec de un persoana
                                 avec chaine double fin
      nr-rec   de 1 a 128
      nr-pers de 1 a 36864
      an       de 1944 a 2100
      luna     de 1 a 12
      zi       de 1 a 31
      motiv    texte (4  60)
     fin
    fin
  ?
```

```
   DESCRIERE FORMATE DE INTRARE PENTRU

    DATELE ASOCIATE UNEI PERSOANE

                (fisierul <MACPER>)

 d
 /*                                                          */
 /*  descriere format date de intrare in sistem              */
 /* <<<     adaugare la structura                   >>>      */
 /*                                                          */
 /* ---------------------------------------------------------*/
 /*    descrierea formatelor pentru incarcare-actualizare    */
 /*      entitate : <persoana>                               */
 /* ---------------------------------------------------------*/
formal pers
 debut
  ident           mot 3
  matricol        mot (13)
  /*------------------------------------------------------*/
  /*    redefinire componente logice                      */
  /*------------------------------------------------------*/
     sex              mot        (4  1)
     an-nastere       dilate     (5  2)
     luna-nastere     dilate     (7  2)
     zi-nastere       dilate     (9  2)
  nume                mot        (16  )
  prenume             mot         16
  /* < loc nastere >                                      */
   judet              dilate     ( 2 )
   localit            dilate      6
   cetatenie          mot         1
   nationalitate      mot         1
   stare-civila       mot         1
  /*  <apartenenta politica>                              */
   tip-apart          mot         4
   an-apart           dilate      4
   ll-apart           dilate      2
   zz-apart           dilate      2
  /* <livret-militar>                                     */
   seria-livret       mot         3
   nr-livret          dilate     (   6     )
 /* <<<<<<  format p02 redefinire  p01                    */
  /* situatia militara                                    */
   specialit          dilate     (17  3)
   comisariat         dilate     (20 2 )
   grad               dilate     ( 22 2)
   obligatii          mot         (24   1)
   grupa-sanguina     mot        (25 4 )
  /* <buletin de identitate>                              */
   seria-bi           mot         29 2
   nr-bi              dilate      31 6
   an-emit-bi         dilate      37 4
   denl-circa         mot        (41 20)
   den2-circa         mot        (61 20)
 /*---------------------------------------------------------*/
 /* exercitiu: descrieti un format de intrare pentru        */
 /* restul atributelor asociate entitati <persoana>         */
 /*---------------------------------------------------------*/
  fin
 fin
 ?
```

```
DESCRIERE FORMATE DE INTRARE PENTRU

ENTITATILE DE TIP NOMENCLATOR

    (fisierul <MACNOM>)

d
/*                                         */
/*  continutul fisierului :                */
/*                                         */
/*          <MACNOM>                       */
/*                                         */
/*                                         */
/* ----------------------------------------*/
/*          <<nomenclatoare>>              */
/* ----------------------------------------*/
formal          macheta
 debut
  ident         mot 3
  filler        mot 77
  /*  <nationalitate-cetatenie>            */
  cod-nac       mot (4  1)
  den1-nac      mot (5 25)
  /*  <grade-militare>                     */
  cod-grm       dilate (4  2)
  den1-grm      mot    (6 27)
  /*          <judete>                     */
  cod-jud       dilate (4  2)
  cifk          dilate (6  1)
  den1-jud      mot    (7 15)
  /*          <localitati>                 */
  ref1-jud      dilate (4  2)
  cod-loc       dilate (6  6)
   ref1-loc   dilate (12  6)
  den1-loc      mot    (18 30)
  den2-loc      mot    (48 10)
  /*          <profesii>                   */
  cod-pro       dilate (4  2)
  den1-pro      mot    (6 30)
  den2-pro      mot    (36 30)
  /*          <specialitati>               */
  ref1-fun      dilate  (4  4)
  cod-spe       dilate  (8  4)
  den1-spe      mot    (12 30)
  den2-spe      mot    (42 30)
  /*          <sanctiuni>                  */
  cod-san       dilate  (4  3)
  den1-san      mot     (7  30)
  den2-san      mot     (37 30)
  /*          <functii                     */
  cod-fun       dilate  (4  4)
  den1-fun      mot     (8 30)
  den2-fun      mot     (38 20)
 fin
 /* sfirsit <bloc formal> <macheta>        */
fin
/* sfirsit <bloc> definire / adaugare <d>  */
?
```

Pentru compilarea acestei structuri și a descrierilor vom trata diferențiat modul de efectuare sub sistemul de operare SIRIS/HELIOS și pe minicalculator.

I) Compilarea textelor exprimate în LDD pentru V 1.5 și V 1.6.R (SIRIS/HELIOS).

a) Presupunem că fișierele DEFSTR, MACNOM și MACPER sînt fișiere membre ale zonei partajate Z % PROC aflată pe perifericul ADO. În acest caz vom efectua o definire în prelucrarea „batch processing" astfel:

— lansare compilare și analiză pentru DEFSTR:

1) • XPROC DEFSTR,MOD DEFSTR este o procedură de comenzi ;
2) • MOD 1,1 șterge primul articol și înlocuiește-l cu :
3) • XPROC PROCGEN instalează monitorul SOCRATE xxxBTCH
4) • OPTION CS'CF : 50,DS : 4096 cu acești parametri în execuție
5) • RUN TIME : 999, NL : 50000, AD : 0,0 lansează monitorul în execuție
6) % SOCBN : BDDPERS,PN : AVRAM, PW : SOC, AN : 1 cerere de conexiune la BDDPERS
7) % RUN FN : D apel compilator LMD
8) • ENDMOD sfîrșit procedura de comenzi

— lansare compilare și analiză MACNOM și MACPER :

• se înlocuiește prima comandă 1) din secvența precedentă cu XPROC MACNON,MOD și respectiv

• XPROC MACPER,DEF ;

• se inserează între comenzile 6) și 7) comanda % ACTI TB : 49 (adăugare la structură) ;

• se lansează procedura respectivă obținută în lucru.

PROCGEN este un fișier de comenzi care realizează instalarea unei componente. Structura acestui fișier de comenzi (procedură catalogată) este prezentată ulterior.

Semnificația comenzilor SIRIS/HELIOS prezente în secvență este redată în manualul de prezentare a sistemului de operare respectiv.

Din punctul de vedere al limbajului de comandă al SGBD SOCRATE pentru prelucrarea în „batch processing" am utilizat aici trei comenzi și anume % SOC, % RUN și % ACTI. Acestea vor fi explicate în detaliu la prezentarea limbajului de comandă pentru prelucrarea „batch".

b) dacă fișierele sînt stocate în spațiul bazei cu ajutorul bibliotecarului de texte atunci după efectuarea procedurii de LOGIN se execută secvența (acțiunile utilizatorului sînt subliniate).

```
$ ACTI TB : 49⟨cr⟩        /* activează adăugarea la structură
$ GO D⟨cr⟩                /* apel procesor LDD
QESTION : $ ⟨cr⟩          /* apel interacțiune procesor LDD-editor
— *I/DEFSTR/⟨cr⟩          /* recuperare text de definire DEFSTR
— *⟨cr⟩                   /* lansare în execuție analiza compilator
QUESTION : $⟨cr⟩          /* apel interacțiune compilator LDD-editor
— *I/MACNOM/⟨cr⟩          /* recuperare text de definire MACNOM
— *⟨cr⟩                   /* compilare efectivă
QUESTION : $⟨cr⟩          /* apel interacțiune compilator LDD-editor
— *I/MACPER/⟨cr⟩          /* recuperare text de definire MACPER
— * ⟨cr⟩                  /* compilare efectivă
QUESTION : NON ⟨cr⟩       /* renunț la procesorul LDD
$ DACT TB : 49⟨cr⟩        /* înhibare adăugare la structură.
```

c) dacă fişierele nu sînt stocate în spaţiul bazei de date atunci pot fi introduse direct de la terminal tastînd textul la apariţia mesajului QUESTION :. Compilarea efectivă a textului introdus este lansată la detectarea caracterului ? care marchează sfîrşitul logic al unui text SOCRATE.

II) Compilarea textelor de definire pe minicalculator. Presupunem că dicţionarul datelor se numeşte BDDPER iar partiţiile CADRE şi SOCLIB (bibliote a de module reentrante SOCRATE) există şi sînt instalate.

Operaţiile efectuate de utilizator, după efectuarea procedu⁻ˈ i LOGIN sistem, sînt următoarele :

a) instalarea procesor de deschidere a sesiunii de lucru cu >IN› **$ LGI** ;

b) instalarea procesor LDD cu **>INS $ DFS** ;

c) deschidere sesiune de lucru cu >LGI DP1 : [1,100] BDD?ER

d) lansare compilator LDD pentru DEFSTR (tipul fişierului este presupus implicit DEF) :

>DFS TI : = DP1 : [1,100] DEFSTR/CVF

e) adăugare la structură a caracteristicilor de tip formal :

>DFS TI : = DP1 : [1,100] MACNON, MACPER/-SA/CVF

Este admisă şi introducerea directă de la terminal a textului prin exploatarea posibilităţilor oferite de sistemul de calcul gazdă (dirijarea intrării şi ieşirii standard pe acelaşi terminal).

Acest mod de introducere conversaţională este foarte dificil datorită faptului că „ecoul" apare pe terminal şi dublează liniile afişate.

Concluzii

Acest capitol a fost destinat prezentării limbajului de definire a datelor pentru versiunile V 1.5 şi V 1.6 ale SGBD-SOCRATE. La tratarea fiecărei caracteristici sînt date diverse detalii privind spaţiul ocupat, cadrajul cerut, structura spaţiului de memorie etc. În principal aceste detalii sînt necesare pentru o mai bună înţelegere a unei anumite implementări şi a filozofiei generale a sistemului.

Mai mult, ele formează un real ajutor pentru ABD în momentul în care acesta doreşte să definească o nouă structură, să optimizeze o structură şi să redefinească o structură.

Forma sintactică generală a caracteristicilor este valabilă pe orice implementare a SGBD-SOCRATE compatibilă cel puţin cu specificaţiile acestor versiuni.

În ultimul paragraf este prezentat un exemplu de definire a unei structuri conceptuale a bazei de date. Structura conceptuală este definită atît prin intermediul reprezentării grafice cît şi cu ajutorul LDD. Pentru structura definită sînt evidenţiate cîteva din operaţiile care se vor efectua asupra bazei de date.

Pentru a putea compila acest text de definire sau textele sale scrise în LDD i se prezintă cititorului modul de realizare a acestei operaţii la implementările care funcţionează pe calculatoare mari şi la implementarea pe minicalculator. Această

prezentare face apel la cunoașterea cel puțin la nivelul „cultură generală informatică" a caracteristicilor limbajelor de comandă ale sistemelor de operare implicate (SIRIS/HELIOS sau MINOS/MIX).

4. LIMBAJUL DE MANIPULARE A DATELOR. LMD-SOCRATE

LMD permite utilizatorului să manipuleze datele stocate în baza de date, invocînd elemente ale structurii sale descrise cu LDD, invocare care nu necesită, comparativ cu limbajele clasice, definirea (redefinirea) formatelor logice ale acestora. Structura este pusă în factor pentru fiecare program sau cerere simplă exprimată în LMD, permițînd atît citarea oricărui element din această structură cît și alegerea unui mod adecvat de acces la acesta. Mai mult, fără a efectua o declarație în acest sens utilizatorul poate părăsi un mod de parcurgere sau acces în favoarea altuia prin simpla formă de exprimare a citării obiectelor. Practic, aceste facilități îi permit să „navigheze" printre valorile asociate obiectelor conform căilor de parcurgere definite în structură sau conform unor căi stabilite dinamic, prin algoritmii de prelucrare.

Programele scrise în LMD pot fi catalogate în spațiul bazei de date într-o formă sursă, adaptabilă la momentul execuției la structura tratată sau direct executabilă pentru o anumită structură.

Aceste programe pot fi scrise de o asemenea manieră încît să funcționeze atît în modul de prelucrare „batch processing" cît și conversațional, alegerea unui mod sau altul însemnînd de fapt simpla lansare în execuție a programului sub controlul unui modul SOCRATE sau altul.

Tipuri de cereri

O *cerere simplă* sau *cerere* este formată dintr-o comandă care determină acțiunea cerută și eventual, o frază care precizează obiectul (obiectele) pe care se desfășoară acțiunea.

Un *program de cereri (program)* este un ansamblu structurat de cereri, terminat printr-un punct de interogare (?).

Programul poate fi structurat pe niveluri, un nivel fiind definit de :

— *un calificator* : {DE... / POUR... FIN}

— *un cuantificator* : {UN/UNE... / TOUT/TOUTE...}

— *o condiție* : SI... FIN

— *un grup de cereri* ALORS : {ALORS... SINON / ALORS... FIN}

— *un grup de cereri* SINON : SINON... FIN
— *o cerere* FAIRE : FAIRE... FIN
— *un filtru* : AYANT sau TELQUE...

Acțiunile și obiectele acțiunii unui program pot fi grupate astfel :
1) acțiuni asupra bazei de date ;
2) acțiuni asupra variabilelor de lucru ;
3) acțiuni asupra programelor ;
4) acțiuni legate de intrări/ieșiri ;
5) acțiuni privind multiprogramarea.

1) *Acțiuni asupra bazei de date*

Orice acțiune asupra bazei de date este subordonată căutării unui element sau al unui (sub)ansamblu al bazei de date. Căutarea unui (unor) element(e) din baza de date se realizează prin :

— desemnarea de ansamblu ;
— metodă de acces ;
— test de selecție ;
— cerere de generare;
— actualizare ;
— suprimare ;
— interogare ;
— numărare ;
— clasare sau inversare a realizărilor entităților.

2) *Acțiuni asupra variabilelor de lucru*

SGBD-SOCRATE pune la dispoziția utilizatorilor un set de ⟨*variabile de lucru*⟩ destinate unor acțiuni specifice.

⟨*variabile de lucru*⟩ : : = ⟨*variabile alfanumerice*⟩/⟨*variabile numerice*⟩/ ⟨*variabile de adresă*⟩
⟨*variabile alfanumerice*⟩ : : = Z_i, $i = 1{,}5$ (7), pentru lucrul cu șiruri de caractere ;
⟨*variabile numerice*⟩ : : = Y_i, $i = 1{,}10$ (25), pentru lucrul cu valori numerice întregi;
⟨*variabile de adresă*⟩ : : = X_i, $i = 1{,}9$, pentru lucrul cu adrese virtuale.

În paranteză a fost indicat numărul maxim de variabile, de tipul respectiv, recunoscute de compilatorul LMD. Utilizarea acestor variabile (Z6, Z7 și Y11 ÷ Y25) este indicat să se facă cu prudență (mai ales în conversațional) deoarece sînt utilizate ca zone de lucru de către unele procesoare.

Variabilele de lucru pot fi utilizate, în funcție de tipul lor, astfel :
— Z_i și Y_i : punere la zi/interogare/test/conversie ;
— Y_i : calcule numerice ;
— Z_i : concatenare șiruri de caractere/extragere subșiruri de caractere ;
— X_i : definire/actualizare/interogare/test.

Numărul variabilelor de lucru poate diferi de la o implementare la alta. În acest caz a fost prezentat setul minim care trebuie pus la dispoziția utilizatorului. Indiferent de modul de implementare aceste variabile sînt manipulate conform destinației asociate.

3) *Acțiuni asupra programelor*

Acest grup de acțiuni se referă la :

— introducerea de cicluri de prelucrare (prelucrări recursive și/sau recurente) cu ajutorul comenzilor :

• POUR... FIN (SUIVANT-SORTIE) ;
• FAIRE... FIN (REFAIRE-SORTIE) ;

— introducere de condiţii : SI...FIN ;

— apel de subprograme : ⟨*nume macro*⟩, EXEC⟨*nume program precompilat*⟩, EXEC ⟨*nume program* IMT⟩.

4) *Acţiuni legate de intrări/ieşiri* :

— rezervarea de buffere în memoria centrală şi formatarea lor conform unor formate logice (de înregistrare externă sau de lucru) ;

— citirea de pe un terminal într-un buffer (inclusiv cititor cartele) ;

— citirea înregistrărilor unui fişier pe supor magnetic (banda sau disc) sau pe un terminal conversaţional ;

— scrierea unui buffer într-un fişier pe suport magnetic (banda sau disc) ;

— citirea unei realizări de entitate într-un buffer formatat conform descrierii entităţii

— scrierea unui buffer formatat conform descrierii entităţii într-o realizare de entitate.

5) *Acţiuni privind multiprogramarea* :

— blocarea şi deblocarea bazei cu care se lucrează (BLOQUER... LIBERER) ;

— blocarea şi deblocarea realizărilor entităţilor cu care se lucrează utilizînd tehnica semafoarelor.

Acest mod de grupare a acţiunilor şi obiectelor acţiunii este relativ în sensul că anumite acţiuni pot avea mai multe obiecte de acţiune.

Principalele comenzi disponibile şi obiectele acţiunii lor sînt specificate în tabela 4.1.

Tabelul 4.1. Principalele comenzi LMD

Obiect acţiune 1) / Comanda	entitate	inel	invers	bloc	referinţe text	cuvînt	lista de valori	valoare numerică	Variabile de lucru X_1	Y_1	Z_1	Acţiune
C	*											Crearea uneia sau mai multor realizări
G	*	*	*									Generarea uneia sau mai multor realizări
S	*	*	*						*			Suprimarea uneia sau mai multor realizări
I	*	*	*	*	*	*	*		*	*	*	Interogare sau imprimare
M	*	*	*	*	*	*	*		*	*	*	Punere la zi
SE			*									Suprimarea unuia sau mai multor elemente ale unui (sub)ansamblu
D									*			Definire

1) Caracteristicile de tip formal pot fi numai obiectul acţiunii comenzii M în versiunea V 1.5

Citaţie

Citaţia reprezintă desemnarea unui element determinat sau a unei caracteristici (obiectul acţiunii precizate de comandă sau atributul citaţiei). Calitatea esenţială pe care trebuie să o îndeplinească această citaţie este aceea de a nu fi ambiguă. Asigurarea acestei calităţi se realizează prin :

– *calificare* (ca în limbajul COBOL), desemnînd filiaţia caracteristicii în conformitate cu structura arborescentă căreia îi aparţine ;

– *cuantificare* utilizînd, înaintea oricărei desemnări de (sub)ansamblu ⟨*entitate*⟩, ⟨*inel*⟩ sau ⟨*invers*⟩), unul din cuantificatorii UN/UNE sau TOUT/TOUTE pentru a preciza dacă interesează unul sau respectiv toţi reprezentanţii ansamblului ;

– *selecţie* cu ajutorul filtrelor (AYANT, TELQUE...) care permite alegerea unui (unor) reprezentant (reprezentanţi) al unui ansamblu, care îndeplineşte (îndeplinesc) anumite condiţii impuse de utilizator.

Calificare. Cuantificare. Selecţie

Calificarea permite evitarea ambiguităţii în raport cu descrierea arborescentă a structurii, şi poate fi :

– *postfixată* (ascendentă), utilizînd cuvîntul rezervat DE ;

– *prefixată* (descendentă), utilizînd cuvîntul rezervat POUR.

Cuantificarea permite evitarea ambiguităţii în raport cu baza de date şi precizează faptul că se va realiza calificarea :

– unui element al unui ansamblu, prin utilizarea unuia din cuantificatorii UN/UNE ;

– tuturor elementelor unui ansamblu, prin utilizarea unuia din cuantificatorii TOUT/TOUTE ;

Identificatorul care urmează cuvintelor cheie DE/POUR se numeşte *calificator*.

Alegerea uneia sau celeilalte modalităţi de calificare este la latitudinea utilizatorului. Diferenţa, între cele două forme este dată de faptul că un calificator desemnat cu POUR este pus în factor (califică) pentru toate citaţiile aflate în „partea stîngă" a comenzilor cuprinse între POUR şi FIN asociat.

Un calificator poate fi :

– o caracteristică (identificatorul caracteristicii) de tip : ⟨*bloc*⟩/⟨*entitate*⟩/⟨*inel*⟩/⟨*invers*⟩ ;

– o variabilă de adresă X_i ;

– cuvîntul cheie XO.

Exemplu :

Dacă se doreşte să se imprime strada domiciliului stabil al unei persoane definită în structură prin STR, se observă că acest identificator este utilizat atît pentru desemnarea străzii domiciliului stabil (în blocul STABIL) cît şi a celui flotant (în blocul FLOTANT). Pentru a desemna, fără ambiguitate, una din cele două informaţii este necesară calificarea acestora, astfel:

– *utilizînd notaţia postfixată* :

```
I STR DE STABIL DE DOMICILIU DE UN PERSOANA ?
I STR DE FLOTANT DE DOMICILIU DE UN PERSOANA ?
```

— *utilizînd notația prefixată* :

```
POUR UN PERSOANA                    POUR UN PERSOANA
 POUR DOMICILIU                     POUR DOMICILIU
  POUR STABIL                         POUR FLOTANT
   I STR                               I STR
  FIN                                 FIN
 FIN                                FIN
FIN                               FIN
?                                 ?
```

Valabilitatea calificatorilor

Pentru a realiza explicarea acestei noțiuni vom face referire, la exemplificări, la noțiuni care vor fi detaliate ulterior.

Regulile definite în acest paragraf sînt reguli de bază ale utilizării calificatorilor în LMD, de aceea am considerat oportună această grupare a lor.

Un calificator definește contextul caracteristicii pe care o califică. Se numește valabilitate contextuală sau mai simplu valabilitate a unui calificator ansamblul caracteristicilor pentru care este validă utilizarea sa.

Deoarece un program de cereri se prezintă ca un ansamblu structurat de cereri, structurare dată de diversitatea posibilităților de grupare a cererilor pe niveluri, un calificator își extinde nivelul de valabilitate de la un nivel la altul conform regulilor (V) următoare :

V1 : la cel mai înalt nivel al programului este valabil implicit contextul „FICHIER" al bazei de date prelucrate. Singurele cuvinte cheie care pot introduce un nou context sînt DE și POUR ;

Exemplu :

Dacă scriem :

```
POUR UN PERSOANA /* pentru prima persoană definită */
   I NUME          /* imprimă numele */
   I STR DE STABIL DE DOMICILIU /* imprimă strada domiciliului stabil */
FIN
?
```

PERSOANA este calificată implicit în contextul „FICHIER". Comanda POUR introduce drept context PERSOANA care va deveni un calificator pentru NUME și DOMICILIU aflate la primul nivel în PERSOANA.

Caracteristica STR este definită în blocul STABIL care, la rîndul său, este definit în blocul DOMICILIU. Cuvîntul cheie DE permite introducerea contextului inglobat conform incluziunii definirilor.

V2 : un calificator anunțat prin *DE* este valabil numai la nivelul caracteristicii pe care o completează ;

Exemplu :

Dacă pentru PERSOANA dorim să imprimăm numele, strada și numărul domiciliului stabil atunci scriem :

```
POUR UN PERSOANA
   I   NUME
   I   STR DE STABIL DE DOMICILIU
   I   NR DE STABIL DE DOMICILIU
FIN
?
```

Deşi STR şi NR au acelaşi calificator la utilizarea notaţiei postfixate este necesară precizarea lui pentru fiecare în parte.

V3 : un calificator anunţat prin *POUR* este valabil pentru toate citaţiile, de cel mai înalt nivel, incluse în grupul POUR...FIN. Acest context este valabil, implicit, pentru fiecare caracteristică de bază a citaţiilor plasate în stînga cererilor ;

{Exemplu :

Referindu-ne la exemplul prezentat la (V2) se observă că citaţiile de cel mai înalt nivel sînt NUME şi DOMICILIU şi acestea sînt calificate implicit prin PERSOANA. Dacă în grupul de cereri am fi avut inclusă şi cererea :
M GRUPA-SANGUINA = 'OI', care reprezintă o atribuire la o listă de valori a unui şir de caractere, atunci GRUPA-SANGUINA, aflată în stînga expresiei de atribuire introduse de comanda M, va fi calificată implicit prin PERSOANA.

V4 : cuvîntul cheie AYANT extinde contextul entităţii pe care o completează la citaţiile plasate în stînga obiectelor criteriilor pe care le introduce :

Exemplu :

Dacă vrem să imprimăm numele tuturor persoanelor născute la data de 01.01.1987 putem realiza acest lucru cu ajutorul unui criteriu de selecţie introdus cu AYANT astfel :

```
    POUR TOUT PERSOANA AYANT
             AN DE DATA-NASTERII = 1987 ET
             LUNA DE DATA-NASTERII = 1 ET
             ZI DE DATA-NASTERII = 1 ;
  I NUME
  FIN
  ?
```

DATA-NAŞTERII este implicit, graţie cuvîntului cheie AYANT, în contextul PERSOANA, iar AN, LUNA şi ZI sînt în blocul DATA-NAŞTERII.

V5 : cuvintele cheie SI, ALORS, SINON menţin contextul nivelului (nivelurilor) de program, care conţin condiţia SI, şi îl introduce la cererile nivelului pe care îl defineşte ;

Exemplu :

Dacă dorim să actualizăm cîmpul PRENUME, pentru toate persoanele care nu au o valoare în acest cîmp, putem realiza un dialog la terminal astfel :

```
POUR TOUT PERSOANA /* pentru toate persoanele */
  SI PRENUME = U /* dacă PRENUME este vid */
    ALORS
      M PRENUME = EXT     /* lansează dialog de atribuire */
                          /* a unei valori pentru PRENUME */
    FIN
FIN ?
```

Cîmpul PRENUME prezent în condiţie (PRENUME=U) şi cel prezent în cererea de atribuire (M PRENUME = EXT) sînt calificate prin PERSOANA.

V6 : cuvîntul cheie FAIRE menţine contextul nivelului care îl conţine la toate cererile definite în cadrul nivelului pe care îl introduce ;

Exemplu:

Dacă pentru o persoană dorim să desemnăm realizările entității calificative prin numărul lor vom scrie:

```
POUR UN PERSOANA
  M Y1=0 /* anulare variabilă de lucru Y1 */
  FAIRE /* ciclu de prelucrare */
    M Y1=Y1 + 1 /* incrementare variabilă cu 1 */
    SI Y1 > 3 /* condiția de ieșire din ciclu */
              /* este parcurgerea a maxim 3 realizări */
              /* ale entității imbricate CALIFICATIVE */
         ALORS SORTIE FIN /* ieși din ciclul definit */
      I UN CALIFICATIVE Y1 /* imprimă conținutul */
      /* caracteristicile definite în realizarea Y1 a entității */
      /* CALIFICATIVE */
    REFAIRE /* reia ciclul de prelucrare */
    FIN     /* sfîrșit ciclu FAIRE */
FIN ?
```

CALIFICATIVE va fi calificat implicit prin contextul PERSOANA introdus de POUR.

V7: se poate realiza o ruptură de context, prin revenire la contextul „FICHIER", în următoarele situații:

- la cel mai înalt nivel al programului;
- la nivelul introdus de TELQUE;
- în orice citare de atribut (partea dreaptă) a unei condiții, filtru, punere la zi sau generare;
- la întîlnirea pronumelui de apel XO.

Exemplu:

Dacă pentru PERSOANA dorim să modificăm numele prin dialog cu utilizatorul putem realiza acest lucru astfel:

```
POUR TOUT PERSOANA
  I  NUME
  I  'DORITI SA-L MODIFICATI (DA, NU) ?'
  M  RASPUNS DE XO=EXT /* preia răspunsul în              */
  /* variabila globală RASPUNS definită la nivel „FICHIER" */
  SI RASPUNS DE XO='DA' /* DE XO realizează o              */
   ALORS                /* ruptură de context              */
    M NUME = EXT        /* deoarece contextul implicit     */
   FIN                  /* este PERSOANA                   */
FIN ?
```

Observație:

Privit prin prisma structurii interne a bazei de date sau făcînd o paralelă cu limbajele de asamblare, un calificator reprezintă o adresă de bază la care se adună adresa relativă a caracteristicii calificate pentru a obține adresa sa reală.

Din analiza reprezentării interne a structurii bazei de date rezultă faptul că toate caracteristicile definite la nivelul blocului „FICHIER" au o adresă care reprezintă o adresă absolută (adresa virtuală de origine). Caracteristicile definite în structuri de tip ⟨*bloc*⟩ sau ⟨*entitate*⟩ au o adresă care reprezintă adresa relativă a acestora față de blocul „tată" (exprimată în cuvinte). În acest context rezultă faptul că, prin calificare, utilizatorul are posibilitatea să „navigheze" într-un sens sau altul, în structura definită pentru baza sa de date. Mai mult orice caracteristică definită la un alt nivel decît „FICHIER" (la acest nivel sînt calificate implicit) trebuie să fie calificată pentru a se putea găsi adresa sa reală.

Calificator de tip ⟨bloc⟩

Sintaxa
{DE/POUR} ⟨*nume bloc*⟩
⟨*nume bloc*⟩ : identificatorul unei caracteristici de tip ⟨*bloc*⟩ în care sînt definite caracteristicile care se califică.

Exemple :

```
I STR /* obiectul acţiunii de imprimare este strada (STR)      */
   DE STABIL /* definit în blocul STABIL                       *
      DE DOMICILIU /* definit în blocul DOMICILIU              */
               DE UN PERSOANA ? /* din entitatea
 PERSOANA aflată la nivel „FICHIER" */
```

sau, în notaţia prefixată :

```
POUR UN PERSOANA ... /* pentru o realizare a entităţii PERSOANA      */
  POUR DOMICILIU .... /* pentru blocul DOMICILIU din PERSOANA        */
    POUR STABIL ....... /* pentru blocul STABIL din DOMICILIU        */
      I STR ............. /* imprimă conţinutul caracteristicii STR  */
    FIN ................. /* terminat valabilitate calificator STABIL */
  FIN ................... /* terminat valabilitate calificator DOMICILIU */
FIN ..................... /* terminat valabilitate calificator PERSOANA  */
?
```

Calificator de tip ⟨entitate⟩/⟨inel⟩/⟨invers⟩

Sintaxa :
{DE/POUR}{UN/UNE/TOUT/TOUTE}{⟨*nume entitate*⟩/⟨*nume inel*⟩/⟨*nume invers*⟩}
⟨*nume entitate*⟩ : identificatorul unei caracteristici de tip ⟨*entitate*⟩ ;
⟨*nume inel*⟩ : identificatorul unei caracteristici de tip ⟨*inel*⟩ ;
⟨*nume invers*⟩ : identificatorul unei caracteristici de tip ⟨*invers*⟩ ;
Un astfel de calificator este utilizat, pentru a desemna fără ambiguitate, astfel :

– ⟨*entitate*⟩ : toate caracteristicile definite la primul nivel în cadrul său ;
– ⟨*inel*⟩ : toate caracteristicile definite la primul nivel în cadrul entităţii care conţine declaraţia de ⟨*referire*⟩ asociată ;
– ⟨*invers*⟩ : toate caracteristicile definite la primul nivel în entitatea căreia îi „inversează" realizările.

Exemple :

1) *calificator de tip* ⟨*nume entitate*⟩

– imprimarea numelui primei persoane definite în baza de date (cuantificator UN/UNE)

```
I NUME DE UNE PERSOANA ?
POUR UN PERSOANA I NUME FIN
```

– imprimarea numelui tuturor persoanelor definite în baza de date (cuantificator TOUT/TOUTE) :

```
I NUME DE TOUT PERSOANA ?
POUR TOUTE PERSOANA I NUME FIN ?
```

— imprimarea numelui și prenumelui tuturor persoanelor definite în baza de date în ordine alfabetică :

```
POUR TOUT PERSOANA X1 PAR NUME;
/* caracteristica ⟨NUME⟩ :: = cheie de acces rapidă *
  I (1) NUME /* imprimă din coloana 1              */
  I (+1) PRENUME  /* imprimă cu o poziție la dreapta */
  ECRIRE          /* față de ultimul caracter diferit */
FIN ?             /* de spațiu sau nedefinit al unui NUME */
```

— imprimarea tuturor calificativelor persoanei cu codul : '1521001031237' :

```
I CALIFIC DE TOUTE CALIFICATIVE /* entitate imbricată */
  DE UN PERSOANA AVEC COD = '1521001031237' ;
?
```

2) *calificator de tip ⟨nume inel⟩* :

— imprimarea numelui tuturor copiilor primei persoane definite în baza de date :

```
POUR UN PERSOANA X1          /* pentru o persoană */
  POUR TOUT PERS-COPII       /* pentru toate realizările COPII   */
    I NUME                   /* grupate de inel PERS-COPII       */
  FIN                        /* imprimă numele                   */
FIN ?
```

3) *calificator de tip ⟨nume invers⟩* :

— imprimarea numelui și prenumelui muncitorilor membrii de partid din anul 1945 :

```
POUR TOUT MUNCITORI /* caracteristica de tip ⟨invers⟩ */
  AYANT AN DE DATA-PCR = 1945; /* filtru de selecție */
  DE XO /* MUNCITORI este definit la nivel „FICHIER" */
        /* iar această declarație este gratuită deoarece */
        /* este modul implicit de calificare */
I (1) NUME
I (+1) PRENUME ECRIRE FIN ?
```

Observație :

Pentru acest grup de calificatori prezența unuia din cuantificatorii {UN/UNE} sau {TOUT/TOUTE} este imperios necesară deoarece calificatorii desemnează un ansamblu sau subansamblu al bazei de date (de la 0 la $n \neq 0$ realizări de entitate).

Calificator de tip ⟨referire⟩

Sintaxa :

{DE/POUR} ⟨*nume referire*⟩

⟨*nume referire*⟩ : identificatorul unei caracteristici de tip ⟨*referire*⟩ ;

Un astfel de calificator este utilizat, pentru a desemna fără ambiguitate, astfel :

— ⟨*referire simplă*⟩ : caracteristicile definite la primul nivel în entitatea căreia îi este asociată.

Exemplu :

```
POUR UN PERSOANA X1
  I PRENUME DE SOT /* caracteristică de tip ⟨referire simplă⟩ */
FIN ?              /* pe entitatea PERSOANA                    */
```

pentru prima persoană definită în baza de date imprimă prenumele soţului (soţiei), dacă caracteristica SOŢ conţine o adresă a unei realizări PERSOANA ;
— ⟨*referire cu inel*⟩ : caracteristicile definite la primul nivel în entitatea care conţine declaraţia de ⟨*inel*⟩ căreia îi este asociată.

Exemplu :

```
POUR UN PERSOANA 10 /* imprimă, dacă există legătura     */
  POUR PERS-PRO X1  /* PERS-PRO, numele profesiei        */
    I DEN1          /* persoanei 10 din baza de date     */
  FIN               /*                                   */
FIN ?
```

Calificator de tip variabila $\langle X_i \rangle$

Variabilele de tip X_i permit „marcarea" unei entităţi într-o citaţie. Marcarea se face prin memorarea adresei entităţii care utilizează variabila, la adresa i a unei tabele de memorie. La momentul detectării (la baza unei citaţii) unei referiri „C DE X_i" se ia drept tată al lui C adresa memorată în locaţia i a tabelei.

Pentru a putea utiliza o variabilă X_i drept marcator este necesar :

— să fie definită formal sau adjectiv (*adjectiv de desemnare sau calificator*) adică să fie identificat cu o clasă de entitate) ;

— să i se desemneze real o adresă (*pronume de apel*) adică să fie identificat cu un reprezentant al entităţii.

Utilizarea variabilelor de tip X_i presupune respectarea următoarelor reguli (R) :
R1 : o variabilă de tip X_i nu poate fi utilizată ca adjectiv decît după un identificator de tip ⟨*entitate*⟩/⟨*inel*⟩/⟨*invers*⟩/⟨*referire*⟩ astfel :
— după ⟨*entitate*⟩ desemnează realizarea entităţii curente.

Exemplu :

```
I NUME DE UN PERSOANA X1 ?
```

imprimă numele primei persoane din baza de date ;
— după ⟨*inel*⟩ desemnează realizarea curentă a entităţii pe ⟨inel⟩.

Exemplu :

```
M X1 = UN FUNCŢII AVEC COD = 10100001 ;
I (1) DEN1 DE XI
I (+0) DEN2 DE X1
  ECRIRE
POUR X1 /* pentru funcţia cu adresa în X1 */
  POUR TOUT FUN—PERS X2 /* pune în X2 adresa fiecărei */
    I (5) NUME          /* realizări a caracteristicii de */
    I (+1) PRENUME      /* tip ⟨referire⟩ asociată şi */
    ECRIRE              /* pentru această adresă */
  FIN                   /* imprimă caracteristicile */
FIN                     /* solicitate din entitatea PERSOANA */
?
```

Pentru funcţia cu codul 10100001 imprimă denumirea, iar pentru toate persoanele (FUN-PERS) imprimă numele şi prenumele ;
— după ⟨*referire*⟩ desemnează realizarea entităţii referire.

Exemplu :

```
I NUME DE COPII-PERS X1 DE UN COPII X2
I (10) NUME DE X2 /* X2 conține adresa unei realizări */
I (+2) PRENUME DE X2 ECRIRE ? și a entității COPII */
```

Imprimă numele tatălui primului copil generat în baza de date iar pentru acesta scrie, începînd cu coloana 10, numele și prenumele pe suportul de ieșire.

— după ⟨*invers*⟩ desemnează realizarea entității curente pe ⟨invers⟩.

Exemplu :

```
M X1=UN MUNCITORI        /* va desemna prima realizare */
I NUME DE X1 ?           /* inversată a entității PERSOANA */
```

imprimă numele primei persoane care este muncitor.

R2 : o variabilă X_i nu poate fi utilizată drept calificator decît după ce a fost definită logic ca adjectiv, adică la execuție definirea ca adjectiv a lui X_i trebuie să preceadă utilizarea sa cu toate că la compilare ordinea poate să difere.

Exemplu :

```
I AN DE DATA DE UN CALIFICATIVE AYANT
CALIFIC=CALIFIC DE SUIVANT DE X1; /* X1 utilizat înainte de definire */
DE UN PERSOANA X1 ? /* aici X1 va fi definit logic */
/* ca adjectiv de desemnare */
```

Pentru prima persoană din bază (UN PERSOANA X1) imprimă anul acordării unui calificativ egal cu calificativul persoanei următoare din bază (SUIVANT).

R3 : o variabilă X_i poate fi definită integral printr-o cerere de punere la zi (M)

Exemplu :

```
M X1 = UN PERSOANA ?
```

X1 desemnează formal și real prima persoană definită din baza de date ;

```
M X2=UN CALIFICATIVE DE UN PERSOANA ?
```

X2 desemnează formal și real prima realizare a entității CALIFICATIVE imbricată în prima realizare a entității PERSOANA.

R4 : o variabilă X_i poate fi definită formal, fără punere la zi simultană, prin cererea de definire.

```
D X1=UN PERSOANA
```

Prin această cerere variabila X_i este pusă la valoarea zero binar (deci fără o desemnare reală) și este definită formal prin identificarea cu o entitate.

Acest tip de definire formală este utilă pentru definirea variabilelor de tip X_i la un nivel superior unei condiții SI.

Punerea la zi a variabilei (desemnarea reală) se poate face pe oricare din ramurile condiției și odată definită poate fi utilizată în toate cererile ramurii respective.

Exemple :

```
D X1 = UN PERSOANA   /* desemnează realizări ale entității   */
M Y1 = EXT           /* PERSOANA                              */
SI Y1 = 1
```

```
ALORS
  .
  .
  .
  M X1 = UN PERSOANA AYANT SEX='B' ;
  . /* adresa primei persoane definită de sex=B */
  .
  .
SINON
  .
  .
  .
  M X1 = UN PERSOANA AYANT SEX='F';
  . /* adresa primei persoane definite de sex=F */
  .
  .
FIN
  .
  .
  .
  ?
```

R5 : o variabilă X_i poate primi, după o definire, o desemnare reală nedefinită (nu și formală) prin cererea M X_i = U

Exemplu :

```
M Y1 = EXT;
M X1 = UN PERSOANA AVEC COD=Y1;
I NUME DE X1
M X1 = U ?
```

R6 : o variabilă X_i poate primi simultan o desemnare formală și reală nedefinită prin cererea D X_i. Această desemnare permite redefinirea formală a variabilelor de tip X_i.

Exemplu :

```
M X1=UN PERSOANA
.    (cereri specifice pentru PERSOANA)
.
D X1 /* anulare variabilă X1 */
FAIRE
  M Y1= EXT
M X1 = UN LOCALITATI AVEC COD  Y= 1 ;
.
.
. (cereri specifice pentru LOCALITATI)
FIN ?
```

R7 : un calificator X_i nu poate fi urmat de nici un alt calificator. Dacă, în notația prefixată, un calificator înglobează un X_i acest calificator își pierde valabilitatea la nivelul citației calificate de X_i.

Exemplu :

```
M X1=UN JUDETE AVEC COD = 1 ;
POUR X1
POUR TOUT JUD-LOC X2 /* JUD-LOC introduce contextul LOCALITATI */
I (1 30) DEN1 /* DENUMIRE LOCALITATE */
  I (+5 30) DEN1 DE X1 /* DENUMIRE */
  I (+0)    DEN DE X1 /* JUDET     */
FIN
FIN ?
```

R8 : pentru un utilizator dat toate variabilele X_i sînt formal și real nedefinite la momentul efectuării procedurii de LOGIN (conectare la baza de date). Utilizatorul le poate aduce la acest stadiu inițial prin comanda D X_i.

Pentru a construi programe independente de prelucrările anterioare este indicat ca fiecare program să pună la nedefinit variabilele X_i atît la începutul programului cît și la terminarea acestuia. Acest lucru permite apelul în lanț al programelor în cadrul aceleiași sesiuni de lucru.

R9 : pentru un utilizator dat valorile atribuite variabilelor X_i, definite la cel mai înalt nivel al unui program rămîn accesibile pînă la efectuarea procedurii de LOGOUT, adică variabilele X_i pot fi conservate de la o cerere la alta (pot fi modificate prin D X_i și utilizate cu o nouă definire formală).

Exemple :

```
⟨apel procesor limbaj de cereri⟩ (ex. RUN FN:R), urmat de :
M X1=UN  PERSOANA 5
I NUME DE X1 ? /* numele persoanei 5 */
⟨apel procesor LMD⟩
I PRENUME DE X1 ?  /* prenumele persoanei 5 */
D X1 ?             /* anulare context variabila X1 */

⟨apel procesor LMD⟩
M X1=UN  LOCALITATI  /* definirea lui X1 cu un */
.                    /* alt context
.
.
```

R10 : o variabilă X_i poate fi redefinită de manieră formală definită numai la același nivel sau la un nivel superior al programului ; în acest caz este valabilă ultima definire.

Exemplu :

```
  ⟨apel procesor LMD⟩
M Y1=EXT  /* lui Y1 i se atribuie o valoare în conversațional */
SI EXISTE UN PERSOANA X1 AVEC COD=Y1 ;
ALORS
   I (1 20) NUME DE X1
   I (23 20) PRENUME DE X1
   ECRIRE
SINON
   M X2=SUIVANT DE X1
   M Y1=COD DE X2
   M X1=UN PERSOANA AVEC COD=Y1 ;
   M Y2=NUMDE X1 I (15–10) Y2 ECRIRE
FIN
D X1 M X1=UN  LOCALITATI DE X0 ?
```

R11 : o variabilă X_i căreia nu i se modifică definirea formală poate fi utilizată la orice nivel al programului mai mic sau egal cu cel în care a fost definită pentru prima dată.

Exemplu :

```
⟨apel procesor LMD⟩
D X1=UN  PERSOANA
D X2=UN  LOCALITATI
```

```
SI EXISTE LOC-STAB X2 DE STABIL DE DOMICILIU DE X1
  ALORS
    POUR X1
      I (1 16) NUME
      I (+1 16) PRENUME
      ECRIRE
    FIN
    I (16 24) 'CU DOMICILIUL STABIL IN:'
    I (+1 30) DEN1 DE X2
SINON
    I (1 12)      '* PERSOANA:'
    I (14 16)     NUME DE X1
    I (31 16)     PRENUME DE X1
    I (+3)        '...NU ARE DOMICILIUL STABIL ??'
FIN
D X1 D X2 ?
```

Observație :

Deoarece redefinirea contextuală a unei variabile de tip Xi este admisă numai la același nivel sau la un nivel superior definirii sale anterioare, utilizatorul poate crea fictiv același nivel introducîndu-se definirea în cicluri „FAIRE...FIN" imbricate conform necesităţil. r sale.

Exemplu :

```
M Y1=EXT
FAIRE
  POUR UN LOCALITATI X1 AVEC COD=Y1 ; FIN
FIN
SI EXISTE Y1 ET Y1 >100
  ALORS
    I NUME1 DE X1
  SINON
    POUR UN JUDETE AVEC COD=Y1 ;
      I NUME1
  FIN
FIN
?
03
```

Calificator XO

Cuvîntul cheie XO desemnează întotdeauna blocul „FICHIER" al bazei de date. Calificarea introdusă de XO este, în general, implicită. Acest calificator este utilizat pentru realizarea unor rupturi de context.

În cazul în care, într-un context introdus prin POUR/AYANT/AVEC/TELEQUE utilizăm caracteristici din alte entități acestea vor fi calificate prin specificarea entității (entităților) de care aparțin calificată(e) (entitatea aflată la nivel „FICHIER") prin DE X0, realizînd astfel o ruptură de context. Rupturile de context sînt valabile numai la nivelul caracteristicilor pe care le completează ;
De exemplu, pentru persoana 10 din baza de date se va imprima numele dacă este muncitor.

Exemplu :

```
<apel procesor LMD>
POUR UN PERSOANA X1 10
  SI EXISTE UN MUNCITORI X2 10 DE X0
```

```
  ALORS
    I NUME
  SINON
    I 'NU ESTE MUNCITOR'
FIN
FIN ?
```

Condiţii

Condiţiile permit exprimarea aserţiunilor logice de prelucrare specifică a datelor conform cerinţelor algoritmilor stabiliţi de utilizator.

Sintaxa de definire a cererilor condiţionale

⟨*condiţie*⟩ : : = ⟨*condiţie simplă*⟩/⟨*condiţie de existenţă*⟩/
⟨*condiţie*⟩ {ET/OU} ⟨*condiţie*⟩

⟨*condiţie simplă*⟩ : : = ⟨*membrul stîng*⟩⟨*operator de relaţie*⟩⟨*membrul drept*⟩

⟨*membrul stîng/drept*⟩ : definesc obiectele comparaţiilor (condiţiilor)

⟨*membrul stîng/drept*⟩ : : = ⟨*cuvînt*⟩/⟨*lista de valori*⟩⟨*valoare numerică*⟩/
⟨*referire*⟩/⟨*inel*⟩/⟨*invers*⟩/⟨*entitate*⟩/
⟨*variabilă* X_i⟩/⟨*variabilă* Y_i⟩/⟨*variabilă* Z_i⟩

Membrul drept admite în plus următoarele tipuri de obiecte ale comparaţiei :

– valori numerice imediate ;
– valori alfanumerice imediate ;
– constanta U (nedefinit).

Exemple :

— **listă de valori comparată cu o valoare alfanumerică :**
SI STARE-CIVILA='1' ALORS ... FIN

— **comparare a două liste de valori :**
SI CETATENIE=NATIONALITATE ALORS ... FIN

— **condiţie compusă :**
SI NUME='POPESCU' ET PRENUME='GHEORGHE' ...

— **comparare valoare numerică :**
SI AN DE DATA-PCR <AN DE DATA-NASTERII
ALORS ... SINON ... FIN

— **comparare cu constanta U :**

```
SI NUME]=U ET PRENUME]=U
  ALORS
    SI SANCT-POL=U OU APARTENENTA = 'NM'
      ALORS
        .
        .
      SINON
        .
        .
        .
    FIN
FIN
```

⟨operator de relaţie⟩ : : ≐ = egal ;
¬ = diferit ;
< mai mic ;
< = mai mic sau egal ;
> mai mare ;
> = mai mare sau egal.

Sinoptic comparaţiile admise, între tipul obiectelor acţiunii unei condiţii, se redau în tabelul 4.2.

Tabelul **4.2. Comparaţii admise între tipurile de obiecte manipulate cu LMD**

membru stîng \ membru drept	cuvînt	listă de valori	constantă alfa-numerică	Z_1	valoare numerică	constantă numerică întreagă	Y_1	referire	X_1	entitate	inel	invers	constantă U
cuvînt	*	*	*	*									*
listă de valori	*	*	*	*									*
Z_1	*	*	*	*			*						*
valoare numerică					*	*	*						*
Y_1				*	*	*	*						*
referire								*	*	*	*	*	*
entitate								*	*	*	*	*	*
inel								*	*	*	*	*	*
invers								*	*	*	*	*	*
X_1								*	*		*	*	*

⟨condiţie de existenţă⟩ : : = {EXISTE/PAS} *⟨citaţie caracteristică⟩*/
{EXISTE/PAS} *⟨variabilă de lucru⟩*

Tipul caracteristicilor acceptate ca obiect al citaţiei sînt :
⟨cuvînt⟩, *⟨valoarea numerică⟩*, *⟨listă de valori⟩*, *⟨entitate⟩*, *⟨inel⟩*, *⟨invers⟩*.
EXISTE (există) şi **PAS** (nu există) sînt complementare.

O cerere condițională poate fi exprimată în una din formele :

SI ⟨*condiție*$_1$⟩ ALORS ⟨*cerere 1*⟩ SINON ⟨*cerere 2*⟩ FIN	sau	SI ⟨*condiție 2*⟩ ALORS ⟨*cerere 2*⟩ SINON ⟨*cerere 1*⟩ FIN

unde ⟨*condiție 2*⟩ este negația lui ⟨*condiție 1*⟩.

Utilizarea operatorilor logici ET (și) și OU (sau) permite descrierea unor expresii condiționale de două sau mai multe condiții.

Evaluarea cererilor condiționale

Orice condiție simplă sau de existență este evaluată la una din valorile ADEVĂRAT (A), FALS (F) sau NEDEFINIT (N).

Deoarece cele două forme de exprimare a unei cereri condiționale dau același rezultat este necesar ca acțiunea lor să fie aceeași în cazul în care condiția este evaluată la nedefinit. Acest lucru implică faptul că prelucrarea unei condiții evaluate la NEDEFINIT nu poate fi asociată nici prelucrării ALORS (condiția evaluată la ADEVĂRAT) nici prelucrării SINON (condiția evaluată la FALS). Valoarea nedefinit a unei condiții duce la non-execuția cererii condiționale (similar cu un salt necondiționat peste o secvență de program).

Evaluarea condițiilor de existență

Modul de evaluare a condițiilor de existență este următorul :

— pentru caracteristicile de tip ⟨*entitate*⟩, ⟨*inel*⟩, ⟨*invers*⟩ condiția de existență introdusă prin **EXISTE** are valoarea A dacă există realizarea citată a caracteristicii respective și F în caz contrar ;

— pentru celelalte caracteristici condiția introdusă prin EXISTE este evaluată la A dacă valoarea asociată caracteristicii este diferită de nedefinit și F în caz contrar. Pentru aceste caracteristici evaluarea condiției de existență este identică cu evaluarea unei comparări a caracteristicii cu constanta *U*.

Exemplu :

```
M X1=UN PERSOANA 15
SI EXISTE NUME DE X1 ALORS ... FIN este echivalentă cu :
SI NUME DE X1=U  ALORS ... FIN ;
```

— existența sau non-existența unei variabile de lucru (Xi, Yi, Zi) este echivalentă unei comparări a acesteia cu constanta *U*.

Exemplu :

```
SI EXISTE Zi ALORS   ... FIN ≡ SI Zi ¬=U ALORS ... FIN
SI PAS Yi ALORS      ... FIN ≡ SI Yi ¬=U ALORS ... FIN
SI EXISTE Xi ALORS   ... FIN ≡ SI Xi ¬=U ALORS ... FIN
```

Evaluarea expresiilor condiţionale

1) expresii condiţionale cu două condiţii :

– intersecţie de condiţii : *ET* (şi)

ET	A	F	N
A	A	F	N
F	F	F	F
N	N	F	N

De exemplu condiţia
NUME ¬= U ET PRENUME ¬= U
se evaluează astfel :
1. se compară conţinutul lui NUME cu valoarea nedefinit :
   1.1. dacă NUME este nedefinit atunci această condiţie este evaluată la A ;
   1.2. dacă NUME este definit atunci este evaluată la F ;
2. se compară ca la 1 conţinutul lui PRENUME ;
3. evaluărilor celor două condiţii li se aplică regulile intersecţiei de condiţii.

– reuniune de condiţii : OU (sau)

OU	A	F	N
A	A	A	A
F	A	F	N
N	A	N	N

De exemplu condiţia compusă
NUME ¬= U OU PRENUME ¬= U
este evaluată conform exemplului precedent (paşii 1 şi 2) după care se aplică regulile reuniunii de condiţii.
Operaţiile logice introduse sînt comutative şi includ logica booleană
2) expresii condiţionale oarecare :
sînt evaluate de la stînga la dreapta ţinînd cont de prioritatea lui ET faţă de OU.

În această evaluare, pentru fiecare cuplu de condiţii, se utilizează regulile enunţate anterior.

De exemplu dacă dorim ca Y1 să ia valori în intervalul [1, 10] ⋃ [30, 40] vom scrie astfel :

```
      ①            ②             ④             ⑤
SI Y1 >= 1  ET  Y1 <= 10  OU  Y1 >= 30  ET  Y1 <= 40
   ③—————————————————————⑦————————————————————⑥
```

Expresia condiţională va fi evaluată astfel :

– se evaluează ① ;

– se evaluează ② ;

– se aplică operatorul ET evaluînd ③ ;

— se evaluează ④ ;
— se evaluează ⑤ ;
— se aplică operatorul ET evaluînd ⑥ ;
— se aplică operatorul OU evaluînd ⑦.

Condițiile simple sau de existență, legate prin operatorii ET sau OU, sînt independente unele față de altele, adică sînt evaluate pe rînd, fără posibilitatea ca evaluarea uneia să atragă după sine reevaluarea celeilalte.

Rezultatul unei expresii condiționate se încadrează în unul din următoarele cazuri :

— în cazul filtrelor introduse cu AYANT sau TELQUE se selecționează o realizare numai dacă rezultatul condiției este ADEVĂRAT ;

— o clauză SINON se execută numai dacă rezultatul condiției este FALS ;

— o clauză ALORS se execută numai dacă rezultatul condiției este ADEVĂRAT.

O evaluare NEDEFINITĂ a unei condiții, în cazul unei citații condiționale, poate fi evitată prin testarea sa (comparare cu constanta U) înaintea condiției.

Exemplu :

Dacă avem o condiție de forma :

```
SI AN DE DATA-NASTERII > 1987
  ALORS
  .
  .
  .
  SINON
  .
  .
  .
FIN
```

atunci : — dacă conținutul lui AN este >1987 se execută ramura ALORS ;

— dacă conținutul lui AN este <= 1987 atunci se execută ramura SINON ;

— dacă conținutul lui AN este nedefinit nu se execută nici una din ramuri.

Dacă dorim să controlăm acest caz nedefinit putem scrie astfel :

```
SI AN DE DATA-NASTERII ] = U
  ALORS
    SI AN DE DATA-NASTERII > 1987
      ALORS
      .
      .
      .
      SINON
      .
      .
      .
    FIN
  SINON
  .
  .    /* cazul AN nedefinit */
  .
FIN
```

Noțiunea de filtru

Limbajul de cereri permite desemnarea uneia sau mai multor realizări ale unui obiect definit în structură. În cazul obiectelor care au o singură realizare calificarea este suficientă pentru a le desemna fără ambiguitate.

În cazul caracteristicilor care pot avea mai multe realizări (⟨*entitate*⟩ ⟨*inel*⟩ sau ⟨*invers*⟩) selecția uneia (unora) din aceste realizări care îndeplinesc anumite condiții se realizează cu ajutorul filtrelor.

Un *filtru* definește criteriile de selecție și/sau modul de acces la un (sub)ansamblu de realizări ale unui ansamblu.

Sintaxa :

⟨*filtru*⟩ : : = ⟨*filtru lent*⟩/⟨*filtru rapid*⟩/⟨*filtru*⟩ ⟨*filtru lent*⟩

⟨*filtru lent*⟩ : : = {⟨*nume entitate*⟩/⟨*nume inel*⟩/⟨*nume invers*⟩} [Xi] [⟨*criteriu sortare*⟩] {AYANT/TELQUE} ⟨*condiție*⟩ ; [⟨*calificare*⟩]

Un ⟨*filtru lent*⟩ (vezi exemplul 1) definește numai condițiile de selecție ale unui subansamblu (parcurgerea realizărilor (sub)ansamblului din care se selectează realizări se face în ordinea secvențială a definirii acestora).

⟨*filtru rapid*⟩ : : = {⟨*nume entitate*⟩/⟨*nume invers*⟩} [Xi] [⟨*criteriu sortare*⟩] AVEC ⟨*condiție simplă*⟩ ; [⟨*calificare*⟩.

Un filtru rapid (vezi exemplul 2) definește atît condițiile de selecție cît și modul de acces la subansamblul selectat (*acces direct*)

⟨*nume entitate*⟩ : identificatorul unei caracteristici de tip ⟨*entitate*⟩ ;

⟨*nume invers*⟩ : identificatorul unei caracteristici de tip ⟨*invers*⟩ ;

⟨*nume inel*⟩ : identificatorul unei caracteristici de tip ⟨*inel*⟩ ;

Pentru citațiile caracteristicilor care reprezintă operanzi în cadrul condițiilor se definesc noțiunile :

– *baza* unei citații este reprezentată de caracteristica aflată la cel mai înalt nivel al bazei de date (FICHIER) la care se ajunge prin calificarea *capului* citației ;

– *capul* unei citații este reprezentat de caracteristica asupra căreia se efectuează citația.

De exemplu, în citația I TOUT PERSOANA AYANT APARTENENTA = 'PCR' ;, care imprimă toate persoanele cu apartenența politică PCR, PERSOANA este baza citației iar capul citației este reprezentat de caracteristica APARTENENTA.

Pentru utilizarea filtrelor (F) se fac următoarele precizări :

F1 : nu este permisă utilizarea calificatorilor între numele unei ⟨*entități*⟩⟨*inel*⟩ sau ⟨*invers*⟩ filtrate și filtru chiar dacă ⟨*entitatea*⟩⟨*inel*⟩ sau ⟨*invers*⟩ filtrată nu este suficient calificată. Calificatorii, dacă există, se plasează după caracterul „;" care închide filtrul (vezi exemplul 2, 4) ;

F2 : calificatorii care apar după închiderea unui filtru pot fi, la rîndul lor, filtrați. Numărul de filtrări succesive este arbitrar (depinde de numărul de imbricări ale caracteristicii de bază) (vezi exemplul 4) ;

F3 : AYANT poate fi utilizat în cazul în care orice bază a condiției este, structural și logic, inclusă în caracteristica filtrată, adică dacă caracteristica filtrată este un calificator al fiecărei baze (vezi exemplul 3) ;

F4 : TELQUE trebuie să fie utilizat atunci cînd o bază a condiției este, structural și logic, exterioară caracteristicii filtrate (vezi exemplul 5) ;

F5 : atunci cînd *baza condiției* este un pronume de apel Xi alegerea între AYANT și TELQUE este la opțiunea programatorului (un pronume de apel Xi rupe

contextul în care se află definită caracteristica indiferent care ar fi citația care o completează) (vezi exemplul 7);

F6 : utilizarea lui AYANT și TELQUE nu este cerută de *capul condiției* nu implică aceeași incluziune pentru capul condiției (vezi exemplul 5);

F7 : în general un filtru urmează imediat numelui de ⟨*entitate*⟩⟨*inel*⟩ sau ⟨*invers*⟩ pe care o filtrează;

F8 : înaintea unui filtru poate apare numai o variabilă Xi și/sau un criteriu de sortare sau acces (vezi exemplele 1 ÷ 8);

F9 : un operator UN exterior unui filtru permite desemnarea primei realizări a bazei condiției care răspunde filtrului (vezi exemplul 2);

F10 : un operator UN interior unui filtru determină analiza tuturor realizărilor bazei condiției pînă la găsirea aceleia care convine (vezi exemplul 6).

Exemple :

1 — **imprimă toate persoanele cu grupa sanguină 'OI' ;**

```
I TOUT PERSOANA AYANT GRUPA-SANGUINA = 'OI' ; ?
```

2 — **imprimă numele și prenumele persoanelor născute în localitatea cu codul 1563171 care au obligații militare de tip '1' :**

```
M X1 = UN LOCALITATI AVEC COD = 1563171;
SI EXISTE X1 ALORS
  POUR TOUT LOCN-PERS X2
          AYANT OBLIGATII DE SITUATIA-MILITARA = '1' ;
          DE X1 /* LOCN-PERS este definit în LOCALITATI */
     I (1) NUME I (+1) PRENUME ECRIRE
FIN FIN ?
```

3 — **numără toți muncitorii (funcția D) membrii P.C.R. cu naționalitatea '1' și starea civilă '2' :**

```
M Y1 = D TOUT MUNCITORI AYANT APARTENENTA = 'PCR'
                                    ET
                              NATIONALITATEA = '1'
                                    ET
                              STARE-CIVILA = '2' ;
I Y1 ?
```

4 — **imprimă anii în care a obținut calificativul 'FB' persoana cu numele IONESCU și prenumele CONSTANTIN născută în anul 1952 :**

```
I AN DE TOUT CALIFICATIVE AYANT CALIFIC = 'FB' ;
   DE UN PERSOANA AYANT NUME = 'IONESCU'
      ET PRENUME = 'CONSTANTIN'
      ET AN DE DATA-NASTERII = 1952 ; DE X0 ?
```

5 — **pentru toate persoanele imprimă numele atîta timp cît RASPUNS este "DA" :**

```
POUR TOUT PERSOANA TELQUE RASPUNS DE X0 = 'DA' ;
   I NUME
   M RĂSPUNS DE X0 = EXT
FIN
```

6 — **imprimă numele tuturor persoanelor care au un calificativ în anul 1987 :**

```
POUR TOUT PERSOANA AYANT AN DE UN CALIFICATIVE = 1987 ;
      I NUME
FIN
```

7 — **exemplul 4 poate fi rescris astfel :**

```
I AN DE TOUT CALIFICATIVE AYANT CALIFIC = 'FB' ;     — —
   DE UN PERSOANA X1 TELQUE
      NUME DE X1 = 'IONESCU' ET
      PRENUME DE X1 = 'CONSTANTIN' ET
      AN DE DATA-NASTERII DE X1 = 1952 ; DE X0 ?
```

Dacă TELQUE se înlocuieşte cu AYANT efectul acestei cereri este acelaşi.
8 — dacă dorim să listăm în ordine alfabetică persoanele cu grupa sanguină '01' vom scrie :

```
POUR TOUT PERSOANA PAR NUME ; /* criteriu de sortare */
        AYANT GRUPA-SANGUINA = 'OI' ; /* filtru */
I NUME FIN ?
```

Noţiunea de acces

Accesul la datele conţinute în baza de date se poate efectua :

- secvenţial ;
- secvenţial ordonat ;
- prin inel ;
- direct.

Acces secvenţial

{UN TOUT}{⟨*nume entitate*⟩/⟨*nume invers*⟩}[Xi]
[{AYANT/TELQUE}⟨*condiţie*⟩] ; ⟨*calificare*⟩]

Accesul secvenţial este accesul de bază asupra datelor conţinute de baza de date. Se realizează prin parcurgerea, în ordinea crescătoare a numerelor de realizare, ale unui ansamblu sau subansamblu de realizări desemnat de un identificator de entitate (inel sau invers). Aceste realizări pot fi selectate eventual, printr-un criteriu. Citaţia de ansamblu sau subansamblu nu trebuie să conţină nici o indicaţie de acces direct sau ordonat. Ordinea de parcurgere corespunde ordinei de generare a realizărilor numai în cazul în care nu au avut loc suprimări ale realizărilor de generări (ordinea este dată de prezenţa biţilor poziţionaţi la 1 în şirul biţilor de prezenţă). Această ordine este o ordine logică în măsura în care ea satisface necesităţile unui utilizator.

Exemple :

— imprimarea numelui şi prenumelui tuturor persoanelor născute înainte de 1945 :

```
POUR TOUT PERSOANA X1 AYANT AN DE DATA-NASTERII < 1945 ;
    I (1) NUME I (+1) PRENUME ECRIRE FIN ?
```

— imprimarea numelui, prenumelui, datei primirii în partid şi a sancţiunii persoanelor din baza de date care au sancţiuni de partid :

```
POUR TOUT PERSOANA AYANT EXISTE SANCT-POL ;
    I (1) NUME I (+1) PRENUME
    I (35) ZI DE DATA-PCR
    I (+1) LUNA DE DATA-PCR
    I (+1) AN DE DATA-PCR
    I (+1) SANCT-POL ECRIRE
FIN ?
```

— imprimarea numelui, prenumelui şi numărului de copii ai persoanelor de sex feminin din baza de date :

```
POUR TOUT FEMEI X1
    M Y1 = D TOUT PERS-COPII DE X1
    I (1) NUME I (+1) PRENUME I (36) Y1 ECRIRE
FIN ?
```

Observaţie : principial, modul de realizare a accesului este :

1 : se determină adresa virtuală a realizării i (∂ar_i) cu formula : ∂ar_i = AVORG + i*LGENTI, unde AVORG este adresa virtuală de origine a primei realizări iar LGENTI lungimea unei realizări ;

2 : se citeste realizarea cu adresa virtuală calculată şi dacă realizarea există atunci :

2.1 : — dacă este definit un ⟨filtru⟩ atunci condiţiile conţinute de acesta se aplică pe valorile citite : dacă valorile satisfac filtrul atunci salt la 2.3 : altfel salt la 1 : ;

2.2 : — dacă nu este definit ⟨filtru⟩ salt la 3: ;

2.3 : — prelucrează realizarea conform cerinţelor utilizatorului ;

3 : salt la 1 : pînă cînd sînt îndeplinite condiţiile de ieşire din ciclu (i = număr maxim de realizări, nu mai sînt realizări definite sau o condiţie impusă implicit de utilizator).

Acces secvenţial ordonat

{UN/TOUT} {⟨*nume entitate*⟩/⟨*nume invers*⟩} [Xi] [⟨*criteriu sortare*⟩]
[⟨*filtru*⟩] [⟨*calificare*⟩]
⟨*criteriu sortare*⟩ : : = PAR ⟨*citaţie cheie*⟩ ;
⟨*citaţie cheie*⟩ : : = ⟨*identificator cheie*⟩ [⟨*calificare*⟩]
⟨*calificare*⟩ : : = DE ⟨*nume bloc*⟩/ DE ⟨*nume bloc*⟩ ⟨*calificare*⟩
⟨*identificator cheie*⟩ : identificatorul unei caracteristici de tip ⟨*cuvînt*⟩, ⟨*lista de valori*⟩ sau ⟨*valoare numerică*⟩ declarată cheie rapidă sau discriminantă la definirea structurii bazei de date.

Acest tip de acces permite parcurgerea unui ansamblu sau subansamblu conform unei ordini impuse de valorile ordonate ale anumitor caracteristici numite caracteristici de sortare (chei de acces rapide sau discriminante).

Exemplu :

Editarea registrului numerelor matricole (numai linia curentă) :

```
⟨apel procesor LMD⟩
POUR TOUT PERSOANA PAR NUME;
I (13 —13) COD
I (14) NUME I (+1) PRENUME
I (51) AN DE DATA-NASTERII
I (+0) '/' I (+0) LUNA DE DATA-NASTERII
I (+0) '/' I (+0) ZI DE DATA-NASTERII
ECRIRE
FIN !
```

Acces prin inel

TOUT ⟨*nume inel*⟩ [Xi] DE {UN/TOUT} ⟨*nume entitate*⟩
[⟨*filtru*⟩] [⟨*calificare*⟩].

Permite parcurgerea unui ansamblu/subansamblu de realizări ale unei entităţi clasat conform unei relaţii de tip inel-referire definită în raport cu un alt ansamblu de realizări.

Exemple :

— **numărarea persoanelor care au aceeaşi funcţie :**

```
POUR TOUT FUNCTII X1
  M Y1 = D TOUT FUN-PERS DE X1  /* numărare realizări grupate */
  I (1) DEN1 I (+0) DEN2        /* pe inel                    */
```

```
I (61) '=' I (70 –6) Y1
  ECRIRE
FIN ?
```

— imprimarea numelui şi prenumelui tuturor copiilor persoanei cu codul '1521001030011' :

```
M Z1 = EXT
POUR TOUT PERS-COPII X1
      DE UN PERSOANA AYANT COD = Z1 ; DE X0
I (1) NUME I (+1) PRENUME
I (+1) RANG-ALOCATIE ECRIRE FIN
?
1521001030011
```

— imprimarea numelui şi prenumelui persoanelor care nu au domiciliul stabil în localitatea de naştere cu codul 030465 :

```
POUR UN LOCALITATI X1 AVEC COD = 30465 ;
   /* condiţia conţine o comparare de adresă */
   POUR TOUT LOCN-PERS X2
      AYANT PERS-LOCN DE DATA-NASTERII ¬ =
                  LOC-STAB DE STABIL DE DOMICILIUL DE X2
      ; DE X2
   I (1) NUME I (+1) PRENUME ECRIRE
   FIN
FIN ?
```

— listarea, în ordine alfabetică, a muncitorilor cu 3 sau mai mulţi copii şi a numărului de copii :

```
POUR TOUT MUNCITORI X1 PAR NUME ;
     AYANT EXISTE PERS-COPII; DE X1
     /* dacă există inelul atunci persoana are copii */
     M Y1 = D TOUT PERS-COPII DE X1 /* număr copii */
     S  Y1 >= 3 /* numărul de copii satisface condiţia ? */
      ALORS
         I (1) NUME I (+1) PRENUME I (+1) Y1
         ECRIRE
      FIN
FIN ?
```

— imprimarea nomenclatorului de localităţi în ordinea crescătoare a codurilor : —

```
POUR TOUT LOCALITATI PAR COD ;
  I (6 –6) COD I (+1 1) '*'
  I (+1 30) DEN1 I (+0 10) DEN2
  ECRIRE
FIN ?
```

Observaţie :

În acest caz parcurgerea secvenţială a realizărilor este abandonată şi se efectuează consultînd o tabelă de indecşi asociată valorilor caracteristicii care constituie *capul* pentru ⟨citaţie cheie⟩. Fiecare intrare din tabelă trimite la realizarea de entitate căreia îi este asociată şi la următoarea intrare a tabelei, următoarea fiind o realizare ale cărei valori îndeplineşte condiţiile de ordonare impuse de utilizator la încărcarea caracteristicii declarate cheie.

Acces direct

Accesul direct se poate realiza prin :

a) număr de realizare ;

b) criterii.

a) *Acces direct prin număr de realizare* :

UN {⟨*nume entitate*⟩/⟨*nume invers*⟩} [Xi] {⟨n⟩ Yi} [⟨*calificare*⟩]

⟨n⟩ : numărul realizării care trebuie să fie selectată ; în cazul utilizării unei variabile Yi aceasta trebuie să fie inițializată, în prealabil, la valoarea ⟨n⟩.

⟨n⟩ ∈ [1, Nmre]

Sistemul calculează adresa realizării ⟨n⟩ cu formula :

∂a r = AVORG + ⟨n⟩ ∗ LGENTI și citește realizarea desemnată de această adresă virtuală. Dacă realizarea desemnată există (este generată) atunci ea este pusă la dispoziția utilizatorului pentru prelucrări ulterioare.

Exemple :

```
— I UN PERSOANA 100 ?
— M Y1 = 100 I UN PERSOANA Y1 ?
— I UN CALIFICATIVE 3 DE UN PERSOANA 15 ?
— I UN MUNCITOR 100 /* deoarece TESA şi MUNCITORI sînt */
  I UN TESA 100 /* disjuncte va imprima PERSOANA 100 */
                /* indiferent de categoria căreia îi */
                /* aparţine */
```

b) *Acces direct prin criteriu*

{UN/TOUT} {⟨*nume entitate*⟩/⟨*nume invers*⟩} [Xi]

AVEC ⟨*criteriu*⟩ [⟨*filtru*⟩][⟨*calificare*⟩]

Se realizează citind în ⟨criteriu⟩ o caracteristică declarată în entitate cheie de acces discriminată (AVEC CLE UNIQUE).

Ansamblul valorilor pe care le ia o astfel de caracteristică este clasat într-un dicționar (tabelă de indecși), fiecare intrare din dicționar punctînd pe realizarea de entitate asociată în mod unic valorii.

Accesul se efectuează astfel :

— se aplică o funcție de „hash-cod" pe valoarea asociată criteriului și se determină o intrare în dicționar ;

— dacă intrarea există atunci punctează spre realizarea de entitate dorită.

Exemple :

— **imprimă persoanele născute între 1965 și 1970 membrii de partid :**

```
  I TOUT PERSOANA X1 AVEC COD >= '1650000000000' ET
    COD =< '1700000000000' ; ET AYANT APARTENENTA = 'PCR' ;
    DE X1 ?
```

— **imprimă nomenclatorul de localități pentru județele cu codul mai mic decît 10 :**

```
  POUR TOUT JUDETE AVEC COD < 10 ;
    I DEN1
    POUR TOUT JUD-LOC X1
    I (6 -6) COD I (+1) DEN1 I (+0) DEN2
    I (56 -6) COD DE SUP-LOC /* acces direct la realizarea aflată      */
    I (+1) DEN1 DE SUP-LOC   /* la nivel ierarhic superior, conform    */
    I (+0) DEN2 DE SUP-LOC   /* organizării administrativ-teritoriale  */
      ECRIRE                 /* accesul este realizat prin referire    */
    FIN
  FIN ?
```

Comenzile (cererile) limbajului de manipulare

În funcţie de interacţiunea sistem-utilizator cererile pot fi grupate în :

– cereri care dau naştere unui dialog între operator (utilizator şi sistem) ;

– cereri independente (nu reclamă intervenţia operatorului în timpul execuţiei).

Pentru cererile care dau naştere unui dialog operator-sistem se va descrie modul de desfăşurare a acestui dialog. Notaţia ⟨cr⟩ utilizată în descrieri semnifică o apăsare pe tasta „RETURN" (sau tasta terminalului care realizează, prin construcţie, această funcţie). În principiu, apăsarea pe această tastă este necesară după orice răspuns sau comandă a utilizatorului.

Funcţia de creare standard şi comanda C (creare)

Deoarece comanda C a fost definită ca o restricţie a funcţiei de creare standard am considerat oportună prezentarea lor împreună. Ambele reclamă un dialog similar cu utilizatorul indiferent de modul de prelucrare ales (batch sau conversaţional). Atît funcţia cît şi comanda C nu au fost încă implementate la Mini.

Funcţia de creare standard

Rol : permite crearea parţială sau totală a bazei de date. Crearea se referă la generarea de realizări ale entităţilor şi atribuirea de valori caracteristicilor definite în aceste entităţi şi/sau baza de date. Utilizarea acestei funcţii nu este obligatorie în sensul că programatorul de aplicaţie dispune de suficiente comenzi care îi permit construirea unor programe de creare a realizărilor de entitate prin care poate efectua prelucrări complexe asupra datelor destinate caracteristicilor (validări logice, construirea unor informaţii derivate, inversare etc.). Această funcţie reprezintă o alternativă comodă oferită utilizatorului şi utilizarea ei este indicată ori de cîte ori satisface necesităţile acestuia. De asemenea, această funcţie reprezintă o unealtă puternică în cazul în care doreşte realizarea unei schimbări majore a structurii bazei de date (datele pot fi salvate pe un fişier cu structura acceptată de această funcţie şi încărcate cu această funcţie în noua bază de date, cu structura modificată).

Această funcţie de creare standard este un procesor al SGBD-SOCRATE, procesor apelabil în prelucrarea conversaţională sau „batch-processing".

a) Utilizarea funcţiei de creare standard în prelucrarea conversaţională.

Apelul procesorului de creare standard se realizează cu comanda :

$GO C ⟨cr⟩

După recepţionarea acestei comenzi sistemul cere utilizatorului să precizeze modul de introducere a răspunsurilor sale (la întrebările sistemului) prin întrebarea :

CREATION CARTE OU CONSOLE ? ⟨*răspuns*⟩⟨*cr*⟩

La furnizarea răspunsului CONSOLE ⟨cr⟩ sistemul :

– parcurge secvenţial structura bazei de date ;

– pentru fiecare caracteristică întîlnită poartă un dialog specific cu utilizatorul.

Pentru caracteristicile de tip ⟨*cuvînt*⟩, ⟨*listă-de-valori*⟩ și ⟨valoare numerică⟩ sistemul anunță :
⟨*identificator*⟩ : și așteaptă unul din răspunsurile :
— /, NON, ⟨cr⟩ : conținutul caracteristicii rămîne neschimbat ;
— U ⟨cr⟩ : caracteristica este pusă la nedefinit ;
— ⟨valoare⟩ : sistemul verifică validitatea valorii (conform condițiilor de validare definite în structură) și în funcție de rezultatul acestei verificări modifică conținutul (valoarea corectă) sau reia dialogul (valoare eronată).

Pentru caracteristicile de tip ⟨text⟩ sistemul cere modificarea sub forma :
⟨identificator⟩ : TEXTE : și așteaptă unul din răspunsurile :
— /, NON : nu se dorește modificarea caracteristicii ;
— OUI : se dorește generarea liniei următoare a caracteristicii
— ⟨valoare⟩ ? : textul ⟨valoare⟩ este introdus în linia generată (sau în linia curentă) ;
— U : caracteristica este pusă la nedefinit ;
— $: apelarea „editorului de texte" SOCRATE pentru utilizarea funcțiunilor acestuia la manipularea liniilor textului (inserare, modificare, ștergere, înlocuire caractere/șiruri de caractere/linii). Sfîrșitul utilizării „editorului de texte" se anunță tastînd comanda **END**. Editorul de texte este prezentat în detaliu în § 5.

Pentru caracteristica de tip ⟨*referire*⟩ sistemul cere modificarea sub forma :
⟨*identificator*⟩ : și așteaptă unul din răspunsurile :
— /, NON, ⟨cr⟩ : referirea nu se actualizează ;
— U : referirea este pusă la valoarea nedefinit și se actualizează legăturile din lanțul căreia îi aparține (păstrează coerența inelului căreia îi aparține) ;
— pe orice alt răspuns sistemul continuă dialogul :

CITATION ; ⟨*citație realizare entitate*⟩ ? unde prin ⟨*citație realizare entitate*⟩ utilizatorul trebuie să desemneze, cu ajutorul LMD, o realizare a entității referite. Cu „?" se marchează sfîrșitul descrierii citației și se lansează în execuție compilatorul LMD. Dacă citația realizării de entitate este corectă, sintactic și sematic, atunci se realizează punerea la zi a caracteristicii și a lanțului de care aparține, altfel se semnalează erorile detectate și se reia dialogul.

Pentru caracteristica de tip ⟨*bloc*⟩ dialogul este :

CREATION ⟨*nume bloc*⟩ ? și se așteaptă unul din răspunsurile :
— OUI : se dorește crearea blocului. Această creare reprezintă de fapt parcurgerea cu dialog a caracteristicilor definite în cadrul său. Dialogul purtat cu utilizatorul depinde de tipul caracteristicilor ;
— /, NON, ⟨cr⟩ : nu se dorește crearea. Pe acest răspuns se realizează un salt peste caracteristicile definite în bloc dialogul continuînd cu prima caracteristică definită în structură la același nivel cu blocul.

Pentru caracteristica de tip ⟨*entitate*⟩ sistemul poartă dialogul :

CREATION (UN/UNE) ⟨nume entitate⟩ ? și așteaptă unul din răspunsurile :
— /, NON, ⟨cr⟩ : nu se dorește crearea unei noi realizări ;
— OUI : se dorește crearea unei realizări a entității ⟨*nume entitate*⟩ ;
— ⟨*n*⟩ : se dorește crearea realizării cu numărul ⟨*n*⟩.

În cazul în care, la cererea de creare a unei noi realizări de entitate, entitatea nu are o realizare (sau realizarea desemnată nu este) disponibilă se semnalează eroarea la terminal iar dialogul continuă cu prima caracteristică, definită la același nivel, din structură.

În cazul validării creării unei realizări sistemul efectuează :

— poziționarea la valoarea 1 a bitului corespunzător realizării create în șirul biților de prezență ;

— un dialog specific pentru fiecare caracteristică definită în cadrul său.

La furnizarea răspunsului **CARTE** sistemul realizează o simulare „batch processing" a prelucrării conversaționale în sensul că răspunsurile la dialogul purtat cu utilizatorul sînt preluate din dispozitivul de intrare standard al partiției în care este lansat SOCRATE. Acest mod de lucru cere ca pe intrarea standard să se găsească un „fișier slash" cu răspunsuri. Acest mod de lucru este indicat numai în cazul în care acest fișier a fost obținut în urma efectuării unor prelucrări automate a datelor deoarece orice eroare sau omisiune a unui răspuns duce în mod implicit la o decalare a acestora și deci la un rezultat eronat.

b) utilizarea funcției de creare standard în prelucrarea „batch processing".

Apelul procesorului de creare standard se realizează cu comanda :

% ⟨*etic* ⟩ RUN FN : C
CARTE
{ răspunsurile
utilizatorului

Dialogul se realizează prin simularea sa pe dispozitivul de ieșire standard al partiției răspunsurile fiind preluate prin dispozitivul de intrare standard al partiției (sau un substitut al acestuia).

Comanda C (creare)

Rol : crearea, în sensul funcției de creere standard, a unei realizări de entitate ;
Sintaxa :

C UN ⟨*citație realizare entitate*⟩ ?

Această comandă determină, după verificarea corectitudinii sintactice și semantice a citației realizării de entitate, un dialog operator-sistem similar celui de creare a unei realizări de entitate :

CREATION (UN/UNE) ⟨*nume entitate*⟩ ?...
[*dialogul cu operatorul*]

unde ⟨*nume entitate*⟩ este identificatorul caracteristicii de tip entitate care constituie capul citației.

⟨*citație realizare entitate*⟩ : : = ⟨*nume entitate* ⟩ [⟨n⟩/Y_1] [⟨*calificare*⟩]

⟨*n*⟩ : numărul realizării pe care dorim să o creem ;

Y_1 :trebuie să i se atribuie anterior utilizării numărul ⟨n⟩ al realizării pe care dorim să o creem.

Dacă numărul realizării nu este specificat atunci se va realiza crearea primei realizări disponibile.

Eventualele erori detectate (realizarea ⟨*n*⟩ a fost creată anterior sau ⟨n⟩ este mai mare decît numărul maxim al realizării posibile sau nu mai există realizări libere) duc la abandonarea prelucrării cu semnalarea tipului și naturii lor.

Pentru caracteristicile definite în entitate dialogul se poartă cu utilizatorul, în funcție de tipul lor, identic cu cel prezentat, pentru fiecare caracteristică, la descrierea funcției de creare standard.

Modul de utilizare a comenzii C în

— prelucrarea conversaţională :

$ GO R ⟨*cr*⟩

QUESTION : C UN ⟨*citaţie realizare entitate*⟩ ? ⟨cr⟩

CREATION (UN/UNE) ⟨*nume entitate*⟩ ? QUI

{ dialog cu utilizatorul purtat în funcţie de tipul caracteristicilor incluse în entitate } ↓ determină

— „batch processing"

% ⟨etic⟩ RUN FN : R

C UN ⟨citaţie realizare entitate⟩ ?

{ OUI
•
• răspunsurile utilizatorului furnizate pe linii de 80 caractere
• cu 1 răspuns/1 linie din coloana 1.

Comanda G (generare)

Rol : permite crearea uneia sau mai multor realizări ale unui (sub)ansamblu.

Modul de realizare şi sintaxa comenzii depinde de tipul (sub) ansamblului.

1) Entitate

Sintaxa :

G UN ⟨*nume entitate*⟩ [X_i] [{⟨*număr*⟩/Y_i}]/TOUT

⟨*nume entitate*⟩ [X_i] [⟨*calificare*⟩]

Specificaţii de utilizare :

Această comandă nu antrenează dialog cu utilizatorul (caracteristicile definite intern entităţii generate sînt lăsate la valoarea nedefinit) ci determină poziţionarea bitului corespunzător (numărului de realizare calculat sau desemnat de utilizator prin ⟨număr⟩/Y_i) din şirul de biţi de prezenţă la valoarea 1. Dacă acest bit era poziţionat la valoarea 1 sau şirul de biţi este complet se semnalează eroare. Dacă se utilizează X_i în comandă atunci acesta devine pronume de apel şi poate fi utilizat ca atare pentru a califica caracteristicile definite în cadrul entităţii pentru eventuale modificări ale acestora.

{⟨*număr*⟩/Y_i} : desemnează realizarea entităţii cu acest ⟨*număr*⟩ (Y_i trebuie poziţionat la valoarea ⟨*număr*⟩ înaintea utilizării sale)

{⟨*număr*⟩/Y_i} $\in$ [1, Nmre] $\cap$ N

Exemple :

— **generarea unei realizări a entităţii PERSOANA urmată de actualizarea cîmpurilor COD, NUME, PRENUME (în conversaţional) :**

```
QUESTION : /* procesor LMD*/
           /* lansat în execuţie */
G UN PERSOANA X1
POUR X1
    M COD = EXT
    M NUME = EXT
    M PRENUME = EXT
FIN ?
```

```
COD : 1521001110031 <cr>
NUME : POPESCU <cr>
PRENUME : ADRIAN-THEODOR <cr>
QUESTION : /* procesor LMD rămîne */
        /* lansat în execuţie */
```

— generarea realizării 501 a entităţii PERSOANA :

```
G UN PERSOANA 501 ?
  sau utilizînd un Yi :
M Y1 = 501
G UN PERSOANA Y1 ?
```

— generarea tuturor calificativelor persoanei 501 :

```
G TOUT CALIFICATIVE DE UN PERSOANA 501 ?
                              calificare
```

2) inel sau invers :

Sintaxa :

G {UN/TOUT} {*<nume inel>*/*<nume invers>*} [*<calificare>*] = *<desemnare realizare(i) entitate>*

Specificaţii de utilizare :

calificare din partea stîngă a relaţiei determină subansamblul *<inel>* sau *<invers>* în care trebuie să se facă generarea iar partea dreaptă determină ralizarea (realizările) care vor fi regrupate. În cazul în care cererea de generare se efectuează asupra unui (sub) ansamblu saturat (numai pentru <invers>) se semnalează eroare pe dispozitivul de ieşire.

Exemple :

— completarea inverselor BARBATI şi FEMEI :

```
—POUR TOUT PERSOANA X1 /* X1 conţine adresa persoanei */
 —SI SEX = 'B' /* conditia de grupare în inversa BARBATI */
    ALORS
      G UN BARBAT DE X0 = X1 /* adresa persoanei */
      SINON
      SI SEX = 'F' /* conditia de grupare în inversa FEMEI */
      ALORS
        G UN FEMEI DE X0 = X1 /* adresa persoanei */
        SINON
           I (1) 'XERP.FF.SEX... PERSOANAI :'
           I (+1) COD ECRIRE
      FIN
    FIN
  FIN ?
```

— regruparea în inversa INVGEN (INVGEN INVERSE TOUT PERSOANA) a tuturor persoanelor care deţin funcţia 12345678 :

```
M X1 = UN FUNCTII AVEC COD = 12345678 ;
G TOUT INVGEN = TOUT FUN-PERS DE X1 ?
```

În acest exemplu FUN-PERS este o caracteristică de tip <inel>.

Comanda S (ştergere)

Rol : permite suprimarea unei realizări de *<entitate>*.

Suprimarea se realizează punînd la valoarea nedefinit toate caracteristicile care aparţin entităţii (inclusiv entităţi imbricate) cu menţinerea coerenţei (eventualele caracteristici de tip <inel> conţinute în entitate sînt puse la nedefinit prin parcurgerea inelului şi actualizarea referirilor corespunzătoare).

Bitul corespunzător numărului de realizare, din șirul de biți de prezență este poziționat la 0 iar cel din șirul de biți de suprimare este poziționat la 1 cu modificarea corespunzătoare a contorilor aferenți acestora.

Sintaxa :

S {X_i/⟨*desemnare de realizare de entitate*⟩}

⟨*desemnare de realizare de entitate*⟩ : : = {UN/TOUT} {⟨*nume entitate*⟩/ ⟨*nume invers*⟩/⟨*nume inel*⟩} [⟨*filtru*⟩] [⟨*calificare*⟩]

Specificații de utilizare :

⟨*nume entitate*⟩ : se suprimă realizarea (UN/UNE) sau realizările (TOUT/TOUTE) entități desemnate, eventual și realizările entităților incluse, și se pun la zi caracteristicile de tip ⟨*referire*⟩, ⟨*inel*⟩ sau ⟨*invers*⟩ conținute de realizare.

Exemple : — șterge persoana 25' din baza de date' :

S UN PERSOANA 25 ?

— ștergerea tuturor realizărilor entității COPII care au AN de naștere mai mic sau egal cu 1965 :

S TOUT COPII AYANT AN DE DATA-NASTERII = < 1965; DE X0 ?

⟨*nume invers*⟩ : se suprimă realizarea entității desemnate și se modifică bitul corespondent, în șirul de biți al caracteristicii de tip ⟨*invers*⟩, la valoarea 0.

Exemple :

— șterge toate persoanele grupate în inversa INVGEN născute înainte de anul 1930 :

S TOUT INVGEN AYANT AN DE DATA-NASTERII < 1930 ; ?

— șterge prima persoană „bărbat" din baza de date :

S UN BARBATI ?

⟨*nume inel*⟩ : se suprimă realizarea (realizările) entității desemnate și se actualizează înlănțuirea spre această entitate.

Exemple :

— ștergerea tuturor localităților din județul 64 :

```
POUR UN JUDETE 64
    S TOUT JUD-LOC
FIN ?
```

— pentru persoana 100 șterge toți copii cu NR-COPIL > 2 :

```
M X1 = UN PERSOANA 100
  S TOUT PERS-COPII AYANT NR-COPIL > 2 ;
    DE X1
  ?
```

— variabila X_i :

a) dacă X_i desemnează o realizare de entitate atunci realizarea va fi ștearsă iar variabila X_i va fi pusă la valoarea nedefinit ;

b) dacă X_i desemnează un buffer pe care s-a placat un format de înregistrare logică (formal) atunci se va realiza o suprascriere a bufferului cu spații pe lungimea dată de mărimea caracteristicii formal. Variabila X_i desemnează în continuare același formal logic.

Exemple :

– M X1 = FORMAL MACHETA DANS BUF 1
S X1 ?
– M X1 = UN PERSOANA AYANT NUME = 'POPESCU' ET
PRENUME = 'ION' ; S X1 ?

Observație :

Deoarece la utilizarea comenzii S, pe desemnare de realizare de entitate, ștergerea bitului din șirul de biți de prezență precede ștergerii caracteristicilor componente, din rațiuni de păstrare a coerenței bazei de date în cazul apariției unor incidente hardware, este de preferat să inversăm ordinea acestor operații.

Inversarea ordinei operațiilor se realizează prin punerea punctuală, a caracteristicilor componente, la valoarea nedefinit (comanda M) după care se dă comanda S.

Deoarece convenția de utilizare a bufferelor la SGDB-SOCRATE este aceea de a transfera din exterior (prin cititre) numai valorile diferite de spatiu (păstrînd vechiul conținut pentru cele cărora le corespund spațiu) este indicată, atunci cînd nu dorim să păstrăm anumite valori, ștergerea bufferului cu comanda S Xi înaintea oricărei citiri. Convenția de transferare a valorilor diferite de spațiu a fost introdusă pentru a permite formatarea bufferelor și păstrarea unor cîmpuri cu valori constante (punerea în factor, pentru mai multe înregistrări, a unei (unor) valori asociate cîmpurilor componente).

Exemplu :

```
POUR UN PERSOANA 100
  POUR TOUT PERS-COPII X1 AYANT NR-COPIL > 2 ;
    M NR-COPIL = U / *anulare punctuală caracteristici */
    M NUME = U /* definite în entitate              */
    M COPII-PERS = U /* anulare legătură cu păstrare coerentă inel */
    .
    .
    .
    S X1 /* ștergere bit de prezență */
FIN FIN ?
```

Comanda SE (ștergere (sub)ansamblu)

Rol : suprimarea clasării unei (unor) realizări de entitate dintr-un (sub)ansamblu ⟨*inel*⟩ sau ⟨*invers*⟩.

Sintaxa :

SE ⟨*desemnare de (sub)ansamblu*⟩

⟨*desemnare de (sub)ansamblu*⟩ : : = {UN/TOUT} {⟨*nume inel*⟩/ ⟨*nume invers*⟩} [⟨*filtru*⟩] [⟨*calificare*⟩]

Specificații de utilizare :

⟨*inel*⟩ : se suprimă înlănțuirea spre realizarea desemnată și totodată se pune la zi referirea conținută de această realizare.

Exemple :

– **suprimarea clasării (apartanenței) localităților la județul 64 :**
SE TOUT JUD-LOC DE UN JUDETE 64 ?
– **suprimarea legăturii persoanei 25 cu primul copil din lanțul PERS-COPII :**
M X1 = UN PERSOANA 25
SE UN PERS-COPII DE X1 ?

⟨*invers*⟩ : bitul (biții) corespunzător(i) realizări(lor) desemnate este (sînt) puse(i) la valoarea 0.

Exemple :

— ştergerea inversei BARBAŢI :

SE TOUT BARBATI ?

— ştergerea realizării 70 a inversei FEMEI :

SE UN FEMEI 70 ?

Observaţie :

Faţă de comanda S care antrenează inclusiv ştergerea fizică a realizării (realizărilor desemnate comanda SE permite o operaţie logică de anulare a unei clasări oarecare a acestei(or) realizări.

Comanda M (atribuire)

Rol : efectuarea operaţiilor de atribuire.

Sintaxa generală :

M *⟨stînga⟩* = *⟨dreapta⟩*

Specificaţii de utilizare generale :

— între obiectele acţiunii comenzii trebuie să existe o identitate de tip. În general nerespectarea acestei identităţi este semnalată încă din faza de compilare a programului (nu este semnalată în această fază pentru atribuirile conversaţionale deoarece utilizatorul execută atribuirea la momentul execuţiei programului său) (conform comparaţiilor admise prezentate în tabela 4.2) ;

— principial, ori de cîte ori obiectul acţiunii comenzii din *⟨stînga⟩* sau *⟨dreapta⟩* este reprezentat de un atribut al structurii bazei de date, modul de acţiune al comenzii este :

a) se verifică existenţa acestui (acestor) element(e) în memoria centrală şi dacă nu este (sînt) aici se antrenează procesul de citire a acestuia (acestora) ;

b) se efectuează modificarea cerută asupra elementului din stînga utilizînd valoarea conţinută de elementul din *⟨dreapta⟩* (modificarea efectuată diferă în funcţie de tipul obiectelor acţiunii comenzii) ;

c) dacă procesul de atribuire rezultat aduce caracteristica din *⟨stînga⟩* la valoarea nedefinit este antrenat eventual, procesul de recuperare a spaţiului eliberat (în cazul în care prin această anulare de caracteristică subpagina care o conţine este vidă ;

d) pagina care conţine caracteristica modificată din *⟨stînga⟩* este transferată pe suportul real (scrisă în baza de date) dacă îndeplineşte anumite condiţii de transfer optimal (timp de staţionare fără utilizare în memoria centrală, solicitare de spaţiu de memorie de către alt proces concurent etc.).

- ori de cîte ori, în prezentare, vom întîlni o construcţie de forma *⟨tip⟩* (*⟨cuvînt⟩*, *⟨valoare numerică⟩* etc.) vom înţelege prin aceasta o citaţie a caracteristicii de acest *⟨tip⟩*, citaţie a cărei sintaxă generală este *⟨identificator⟩* [*⟨calificare⟩*], unde *⟨calificare⟩* poate lipsi numai dacă este prefixată (prefixarea este valabilă numai pentru *⟨stînga⟩*) ;

⟨stînga⟩ : : = *⟨cuvînt⟩*/*⟨listă-de-valori⟩*/*⟨valoare numerică⟩*/*⟨referire⟩*/*⟨text⟩*/ *⟨variabile de lucru⟩*/*⟨cuvînt formal⟩*/*⟨zecimal despachetat⟩*/*⟨zecimal împachetat⟩*/*⟨nume macro⟩*

⟨dreapta⟩ : : = EXT/U/*⟨cuvînt⟩*/*⟨listă-de-valori⟩*/*⟨valoare numerică⟩*/*⟨referire⟩*/ *⟨desemnare realizare de entitate⟩*/*⟨variabile de lucru⟩*/*⟨cuvînt*

formal⟩/⟨*zecimal despachetat*⟩/⟨*zecimal împachetat*⟩/⟨*constantă*⟩/
⟨*funcție*⟩/⟨*expresie*⟩/⟨*citație formare buffer*⟩/⟨*nume macro*⟩

⟨*nume macro*⟩ : este numele unui program de tip macroinstrucțiune care conține, definit în modelul de expansiune, unul din elementele care îl preced în lista producțiilor alternative.

Operația de atribuire este considerată, implicit, ca operație de comparare a identității de tip între obiectele acțiunii comenzii. Deoarece modul de acțiune al comenzii este dependent de tipul obiectelor acțiunii vom face o prezentare al acesteia, tratînd fiecare element implicat.

La efectuarea atribuirii se aplică în mod automat, asupra valorii din ⟨*dreapta*⟩, condițiile de validare de la tipul de obiect respectiv (pentru caracteristicile din structura bazei de date vezi capitolul 3).

Ca regulă generală, dacă membrul drept este nedefinit sau constanta **U** atunci conținutul obiectului acțiunii din ⟨*stînga*⟩ va fi șters fizic, ștergerea menținînd coerența logică a eventualelor clasări ale elementului în care este definit.

În continuare vom prezenta regulile de producție pentru constantele de tip numeric și alfanumeric. Aceste reguli și tipuri de constante sînt acceptate ca atare de către toate versiunile mai vechi sau mai noi ale LMD.

Ulterior vor fi prezentate și alte tipuri de constante acceptate la prezentarea versiunii care le-a introdus.

⟨*constantă*⟩ : : = ⟨*constantă numerică*⟩/'⟨*constantă alfanumerică*⟩'
⟨*constantă numerică*⟩ : : = [⟨*semn*⟩] ⟨*număr*⟩
⟨*număr*⟩ : : = ⟨*cifră*⟩/⟨*cifră*⟩ ⟨*număr*⟩
⟨*cifră*⟩ : : = 0/1/2/3/4/5/6/7/8/9
⟨*semn*⟩ : : = +/Ђ/ −
+/Ђ : număr pozitiv ;
− Ђ : număr negativ ;

Exemple :

−1987, +123, −1, 6 etc.

⟨*constantă alfanumerică*⟩ : : ; ⟨*caracter*⟩/⟨*caracter*⟩ ⟨*constantă alfanumerică*⟩
⟨*caracter*⟩ : : = ⟨*cifră*⟩/⟨*literă*⟩/⟨*caracter special*⟩
⟨*literă*⟩ : : = A/B/C/D/E/F/G/H/I/J/K/L/M/N/O/P/R/S/T/U V/W/X/Y/Z
⟨*caracter special*⟩ : : = #/=/</>/¬/−/%/∂/;/./,//Ђ/$/£//'/+/−/*/
⟨*caracter special*⟩ : orice caracter diferit de ⟨*literă*⟩ și ⟨*cifră*⟩ din setul de caractere acceptate de sitemul de calcul.

În această producție au fost prezentate caracterele speciale utilizate curent de majoritatea aplicațiilor.

Exemple :

'EXEMPLU', '%S ¬ = #', '∀' unde ∀ este obținut, de exemplu, prin supra-perforare (12−9−8−1−0, valoarea zero binar EBCDIC) în cazul utilizării „cartelei". În cazul în care ⟨*constanta alfanumerică*⟩ este de forma [⟨*semn*⟩] ⟨*număr*⟩ atunci acest număr va fi reprezentat în formatul zecimal extern (zecimal despachetat) precedat eventual de ⟨*semn*⟩. De exemplu constanta '−123' este reprezentată în EBCDIC astfel

C, 0, F, 1, F, 2, F, 3

⟨*funcţie*⟩ : : = D/NUMDE/SUIVANT DE

1) M ⟨*cuvînt*⟩$_s$ = EXT/U/⟨*cuvînt*⟩$_d$/⟨*cuvînt formal*⟩/Z_1/
'⟨*constantă alfanumerică*⟩'/⟨*lista-de-valori*⟩

Transferul se efectuează de la stînga la dreapta cu respectarea regulilor :

— dacă lungimea rezervată pentru ⟨*cuvînt*⟩$_s$ este mai mică decît a emiţătorului se va realiza o trunchiere dreaptă a acestuia ;

— dacă lungimea rezervată pentru ⟨*cuvînt*⟩$_s$ este mai mare decît a emiţătorului se tranferă integral emiţătorul iar receptorul este completat cu valoarea nedefinit, pe caracterele rămase libere, pînă la completarea lungimii sale.

Dacă comanda este de forma *M* ⟨*cuvînt*⟩$_s$=*EXT* atunci este declanşat un dialog cu operatorul de la terminal astfel (sau simulare în batch) :

⟨*identificator*⟩ : [⟨*răspuns*⟩] ⟨cr⟩ unde ⟨*răspuns*⟩ poate fi :

— /, NON, ⟨cr⟩ : nu se doreşte modificarea caracteristicii desemnate de ⟨identificator⟩ ;

— U : caracteristica este pusă la nedefinit ;

— ⟨*valoare*⟩ : caracterele care compun elementul ⟨*valoare*⟩ sînt transferate, respectînd regulile de transfer (⟨*valoare*⟩ este de tip ⟨*constantă alfanumerică*⟩).

Exemple :

```
— POUR UN PERSOANA X1 10        /* pentru persoana 10 se        */
  M NUME = EXT                  /* atribuie în conversaţional NUME */
  POUR TOUT PERS-COPII          /* şi se propagă pentru toţi copii săi */
    M NUME = NUME DE X1         /* numele de familie
  FIN
  FIN ?
— M Z1 = 'IONESCU'
  M NUME DE PERSOANA 77 = Z1 ?
            sau, echivalent :
  M NUME DE UN PERSOANA 77 = 'IONESCU' ?
```

2) M ⟨*lista-de-valori*⟩$_s$; EXT/U/⟨*cuvînt*⟩/⟨*lista-de-valori*⟩/Zi/
⟨*cuvînt formal*⟩/'⟨*constantă alfanumerică*⟩'

Dacă comanda este de forma M ⟨*lista-de-valori*⟩ = EXT atunci este declanşat un dialog cu utilizatorul (sau se simulează în batch) astfel :

⟨*identificator*⟩ : şi aşteaptă unul din răspunsurile :

— /, *NON*, ⟨cr⟩ : conţinutul caracteristicii rămîne neschimbat ;

— *U* ⟨cr⟩ : caracteristica este pusă la nedefinit ;

— ⟨*valoare*⟩ : sistemul verifică validitatea valorii (apartenenţă la lista) şi în funcţie de rezultatul acestei verificări modifică conţinutul (valoare corectă) sau reia dialogul (valoare eronată).

Exemple :

```
⟨apel procesor LMD⟩
FAIRE /* lansare prelucrare în ciclu */
M Z1 = EXT
SI  Z1 = 'GATA' ALORS SORTIE FIN /* condiţia de ieşire din ciclu */
M X1 = UN PERSOANA AVEC COD = Z1 ;
POUR X1 /* se atribuie automat valori unor caracteristici */
        /* dacă realizarea identificată de conţinutul lui Z1 există */
  SI PAS SEX ALORS M SEX = 'B' FIN
  I SEX M SEX = EXT          /* se imprimă conţinutul */
  SI EXISTE APARTENENTA-POLITICA /* existent al caracteristicilor */
```

```
    ALORS                              /* şi se dă controlul        */
      I APARTENENTA-POLITICA           /* utilizatorului pentru a   */
  FIN                                  /* efectua modificările      */
  M APARTENENTA-POLITICA = EXT
  FIN                                  /* dorite asupra acestora    */
  REFAIRE /* relansează prelucrarea ciclului */
FIN ?
```

```
Z1 : 1530110123450 ⟨cr⟩          |
SEX : B                          |
SEX : ⟨cr⟩                       |
APARTENENTA-POLITICA : ODUS      |  exemplul de dialog la terminal
APARTENENTA-POLITICA : ⟨cr⟩      |
Z1 : GATA                        |
```

Observaţie : dacă realizarea entităţii desemnate nu există atunci nu se va purta dialog cu utilizatorul pe caracteristicile SEX şi APARTENENTA-POLITICA.

3) M ⟨*valoare numerică*⟩ = EXT/U ⟨*valoare-numerică*⟩ *Yi*/ ⟨*zecimal despachetat*⟩/ ⟨*zecimal împachetat*⟩/⟨*constantă numerică*⟩

Dacă comanda este de forma M ⟨*valoare numerică*⟩ = EXT atunci dialogul purtat de utilizator este de forma :
⟨*identificator*⟩ : şi aşteaptă unul din răspunsurile :
– / NON, ⟨cr⟩ : nu se modifică conţinutul caracteristicii ;
– U ⟨cr⟩ : conţinutul caracteristicii este poziţionat la nedefinit ;
– ⟨*constantă numerică*⟩ : sistemul verifică validitatea valorii (apartenenţă la intervalul de variaţie şi test de numericitate) şi în funcţie de rezultatul acestei evaluări modifică conţinutul caracteristicii (valoare corectă) sau reia dialogul (valoare eronată).

Exemple :

La cererea :
M CODNUM DE X0 = EXT
– dacă răspunsul este U atunci conţinutul lui CODNUM va fi şters ;
– dacă valoarea furnizată este –123, de exemplu, atunci va fi refuzată deoarece nu aparţine intervalului ;
– dacă valoarea furnizată este 123 atunci va fi acceptată şi conţinutul lui CODNUM va fi modificat corespunzător acestei valori.

4) M ⟨referire⟩ = EXT/U/X_i/⟨*desemnare realizare de entitate*⟩ ⟨*desemnare realizare de entitate*⟩ : desemnarea unei realizări de entitate (⟨*inel*⟩ sau ⟨*invers*⟩), prin citaţii specifice efectuate cu ajutorul LMD, conformă cu declaraţia de asociere descrisă în structura bazei de date. Dacă membrul drept este Xi, atunci Xi trebuie să conţină adresa unei realizări de entitate (lui Xi i se poate atribui, în acest caz, chiar şi o referire).
Dacă membrul drept este U atunci caracteristica de tip ⟨*referire*⟩ este pusă la nedefinit (dacă caracteristica este de tip ⟨*referire cu inel*⟩ se modifică, înaintea ştergerii, înlănţuirile de clasare a acesteia în inelul asociat).

Dacă comanda este de forma M ⟨*referire*⟩ = EXT atunci dialogul purtat cu utilizatorul este de forma :
⟨*identificator*⟩ : şi aşteaptă unul din răspunsurile :
– / NON, ⟨cr⟩ : referirea nu se modifică ;
– U ⟨cr⟩ : referirea se pune la valoarea nedefinit ;
– pe orice alt răspuns continuă dialogul astfel :

CITATION : ⟨*desemnare realizare de entitate*⟩ ? ⟨cr⟩

Dacă ⟨*desemnare realizare de entitate*⟩ este corectă, din punct de vedere sintactic și semantic, atunci se realizează punerea la zi a caracteristicii, și eventual a lanțului de care aparține, altfel se semnalează erorile detectate și se reia dialogul.

Observație : pentru ca utilizatorul să nu piardă timpul la terminal sau pentru a da posibilitatea unui personal cu pregătire informatică redusă sau nulă să opereze terminalul este indicat ca citațiile pentru ⟨desemnare realizare de entitate⟩ să fie definite ca macroinstrucțiuni, eventual parametrizate.

Exemplu :

Dacă dorim să actualizăm referirea COPII-PERSOANA (COPII-PERS) putem construi un program denumit PARINTE : (:reprezintă poziția sau parametru) al cărui corp conține citația :

UN PERSOANA AVEC COD = ':1:' ; DE X0

În acest caz la actualizarea caracteristicii de tip referire

M COPII-PERS DE UN COPII ... = EXT

dialogul poate fi (răspunsurile subliniate) :

COPII-PERS : **DA** ⟨cr⟩

CITATION : **PARINTE 1531001123456 ?** ⟨cr⟩, altfel citația ar fi trebuit să fie exprimată explicit astfel :

CITATION : **UN PERSOANA AVEC COD = '1531001123456' ; DE X0 ?**

ceea ce ar implica atît un volum mai mare de muncă la operare cît și cunoașterea structurii logice a datelor și a limbajului de manipulare a datelor.

Pentru mai multe detalii privind modul și sintaxa de construire a programelor de tip macroinstrucțiune se va studia capitolul 5.

În cazul schimbării clasării unei realizări a unei entități dintr-un inel în altul este preferabilă o anulare prealabilă a vechii clasări urmată de noua clasare, astfel :

M ⟨*referire*⟩ = *U*

M ⟨*referire*⟩ = ⟨*desemnare realizare de entitate*⟩

Exemple :

```
M COPII-PERS DE UN COPII 100 = UN PERSOANA 1 ?
M X1 = UN PERSOANA AVEC COD = Z1 ; DE X0
M PERS-FUN DE X1 = UN FUNCTII AVEC COD = Y1 ;
M X2 = UN FUNCTII AVEC COD = Y1 ;  /* aceste două operații sînt echivalente */
M PERS-FUN DE X1 = X2               /* ca efect, cu precedenta               */
M X3 = PERS-FUN DE X1               /* atribuirea la o variabilă Xi a unei   */
M PERS-FUN DE X1 = X3               /* caracteristici de tip referire și atribuirea */
                                    /* variabilei Xi la o referire           /*
```

5) M ⟨*text*⟩ = EXT

În acest caz dialogul purtat cu utilizatorul este identic cu cel prezentat în §4.

6) M X_i ; U/X_i/⟨*desemnare realizare de entitate*⟩

⟨*desemnare realizare de entitate*⟩ citirea cu ajutorul LMD a unei realizări de ⟨*entitate*⟩ (⟨*inel*⟩, ⟨*invers*⟩ sau ⟨*referire*⟩).

7) M Z_i=U/EXT

M Y_k = U/EXT

În cazul în care membrul drept este EXT dialogul purtat de utilizator este de forma :

{Y_k/Z_i} : și așteaptă unul din răspunsurile :

— / NON, ⟨cr⟩ : valoarea variabilei rămîne neschimbată ;

— U ⟨cr⟩ : variabilei i se atribuie valoarea nedefinit ;

— ⟨*valoare*⟩ : variabila primește, după validare, parțial sau integral această valoare.

Restul comparaţiilor admise la comanda M sînt prezentate ca paragrafe separate, ale comenzii, datorită importanţei sau formei lor specifice de producere.

Utilizarea funcţiei D (Denombrer)

Rol : numărarea elementelor unui (sub)ansamblu.
Sintaxa :
M Y_i = D TOUT ⟨*desemnare de (sub)ansamblu*⟩
⟨*desemnare de (sub)ansamblu*⟩ : : = {⟨*nume entitate*⟩/⟨*nume invers*⟩/ ⟨*nume inel*⟩} [⟨*filtru*⟩] [⟨*calificare*⟩]
Această comandă este echivalentă cu secvenţa (1) :
M Y_i=0
POUR TOUT ⟨*desemnare de (sub)ansamblu*⟩
  M $Y_i = Y_i + 1$
FIN
Specificaţii de utilizare :
Dacă ⟨*desemnare de (sub)ansamblu*⟩ este nedefinită variabila căreia i se atribuie rezultatul funcţiei primeşte valoarea 0.
Pentru (sub)ansamblu ⟨*entitate*⟩ sau ⟨*invers*⟩ ale cărei realizări nu fac obiectul unui ⟨*filtru*⟩ operaţia de numărare se reduce la simpla transformare a valorii contorului şirului de biţi (de prezenţă pentru entitate) în variabila Y_1.
Dacă ⟨*filtru*⟩ este prezent atunci numărarea se realizează parcurgînd secvenţial şirul de biţi şi selectînd, prin aplicarea valorilor fiecărei realizări la ⟨*filtru*⟩ pentru numărare numai realizările care răspund filtrului.
Secvenţa (1) realizează o parcurgere secvenţială a (sub)ansamblului indiferent dacă există sau nu un ⟨*filtru*⟩. În cazul caracteristicilor de tip ⟨*inel*⟩ funcţia de numărare se realizează parcurgînd secvenţial toate realizările grupate de acest ⟨*inel*⟩. În acest caz secvenţa (1) este echivalentă ca timp de execuţie cu comanda însă oferă avantaje prin posibilitatea controlării ei de către utilizator (inserarea unor secvenţe de control între POUR şi FIN asociat) în sensul abandonării sau continuării prelucrării sale.

Exemple :

— numărarea personalului muncitor din B.D. ;

M Y1 = D TOUT MUNCITORI I Y1

— numărarea persoanelor BĂRBAŢI, căsătoriţi cu unul sau mai mulţi copii :

M Y1 = D TOUT BARBATI X1 AYANT STARE-CIVILA = '1' ET EXISTE PERS-COPII DE X1 ; DE X0

— numărarea persoanelor din B.D. :

M Y1 = D TOUT PERSOANA

— numărarea copiilor unei persoane oarecare :

M Y1 = D TOUT PERS-COPII DE UN PERSOANA

Utilizarea funcţiei NUMDE (Numérs de réalisation/Valeur)

Rol : determinarea numărului de realizare al unei realizări de ⟨*entitate*⟩ (⟨*referire*⟩ ⟨*inel*⟩ sau ⟨*invers*⟩) sau a numărului elementului valoare atribuit unei ⟨*liste de valori*⟩.

Sintaxa :

1) ⟨lista de valori⟩ :

M Y_i = NUMDE *nume* ⟨*lista*⟩ [⟨*calificare*⟩]

2) ⟨*entitate*⟩ sau ⟨*invers*⟩ :

M Y_i = NUMDE {⟨*nume entitate*⟩/⟨*nume inel*⟩}[⟨*filtru*⟩][⟨*calificare*⟩]

3) variabilă de adresă :

M Y_i = NUMDE X_i

Xi : trebuie să desemneze o realizare de entitate.

4) pentru caracteristica de tip ⟨*inel*⟩ :

```
POUR TOUT ⟨nume inel⟩ Xi [⟨filtru⟩][⟨calificare⟩]
    M Yi = NUMDE Xi /* numărul de realizare al
                    /* entităţii care conţine declaraţia
                    /* de referire asociată
FIN
```

5) pentru caracteristica de tip ⟨*referire*⟩

M Y_i = NUMDE ⟨*nume referire*⟩ [⟨*calificare*⟩]

În acest caz numărul de realizare găsit este cel al realizării entităţii care conţine caracteristica de tip ⟨*inel*⟩ asociată (capul de lanţ).

Exemple :

```
POUR TOUT PERSOANA   X1
   M Y1 = NUMDE X1 /* ⟨entitate⟩ */
   M Y2 = NUMDE CETATENIE DE X1 I Y1 I Y2 /* ⟨lista⟩ */
   POUR TOUT PERS-COPII X2 /* ⟨inel⟩ */
      M Y3 = NUMDE X2
      I Y3
   FIN
   M Y4 = NUMDE PERS-FUN DE X1 /* ⟨referire⟩ */
   I Y4
FIN ?
M Y1 = NUMDE UN PERSOANA AVEC COD = Z1; ?
M Y1 = NUMDE UN BARBATI AVEC COD = Z1; ?
M X1 = UN BARBATI AVEC COD = Z1 ;
M Y1 = NUMDE X1?
```

Utilizarea funcţiei SUIVANT DE

Rol : permite accesul secvenţial la ansamblul realizărilor unei entităţi sau la un subansamblu de realizări definit de un filtru, apartenenţă la un inel, apartenenţă la un dicţionar etc. ;

Variantele sintactice și modurile corespunzătoare de utilizare sînt următoarele :

1) M X_i=SUIVANT DE UN ⟨*nume entitate*⟩ [⟨*calificare*⟩]

– X_i trebuie definit formal, în prealabil, ca adjectiv pentru ⟨*nume entitate*⟩. În acest caz X_i punctează pe o realizare de entitate iar accesul se efectuează pe realizarea care urmează imediat celei desemnate de X_i (în ordinea crescătoare a numerelor de realizare) ;

– dacă X_i=U se efectuează acces la prima realizare ;

– dacă X_i nu punctează formal entitatea desemnată se semnalează eroarea de sintaxă ;

– dacă X_i punctează pe o realizare de entitate care nu aparține la clasa de entitate determinată de ⟨*calificare*⟩ X_i este pus la valoarea U.

Exemplu :

```
M X1 = UN PERSOANA /* definire ca adjectiv */
M X2 = UN PERSOANA /* (rămîne valabilă pentru toate exemplele) */
M X2 = SUIVANT DE UN PERSOANA X1
```

X1 dă adresa primei persoane definite iar X_2 adresa primei realizări definite după aceasta.

2) M X_i=SUIVANT DE UN ⟨*nume entitate*⟩ AYANT ⟨*filtru*⟩ ;
[⟨*calificare*⟩]

– X_i punctează pe o realizare de entitate (care răspunde sau nu filtrului) iar accesul se realizează la următoarea realizare care răspunde filtrului ;

– dacă X_i=U se efectuează acces la prima realizare care răspunde filtrului ;

– dacă X_i nu punctează formal pe entitatea desemnată se semnalează eroare de sintaxă ;

– dacă X_i punctează pe o entitate care nu aparține ansamblului realizărilor determinate de calificare atunci X_i este pus la nedefinit.

Exemplu :

```
M X2 = SUIVANT DE UN PERSOANA
       AYANT NUME = 'POPESCU';
```

X2 conține adresa următoarei persoane care are numele POPESCU. Dacă nu există o realizare care îndeplinește condiția impusă atunci va primi valoarea nedefinit.

3) M X_i=SUIVANT DE UN ⟨*nume inel*⟩ [⟨*calificare*⟩]

– dacă X_i punctează pe o realizare de entitate care nu este clasată de inel se va efectua acces la elementul care urmează în inel (în ordinea înlănțuirii) ;

– dacă X_i=U se efectuează acces la primul element definit în inel ;

– dacă X_i nu punctează pe inel atunci este pus la nedefinit.

Exemplu :

```
M X3 = UN COPII
M X3 = SUIVANT DE UN PERS-COPII DE X1
```

X3 va conține adresa realizării COPII. Dacă, de exemplu, algoritmul de introducere este LIFO și este respectat atunci adresa este a penultimului copil.

4) M X_i=SUIVANT DE UN ⟨*nume entitate*⟩ AVEC ⟨*criteriu*⟩ ;
[⟨*calificare*⟩]

– dacă X_i punctează pe o realizare care răspunde criteriului se efectuează acces la realizarea următoare, care răspunde criteriului, în ordinea sinonimelor în dicționar ;

— dacă X_i=U se punctează pe primul element care răspunde criteriului în dicționar;
— dacă X_i nu desemnează o realizare care răspunde criteriului atunci va fi pus la nedefinit.

Exemplu:

M X2 = SUIVANT DE UN PERSOANA AVEC
COD = '1870202030451';

5) M X_i=SUIVANT DE UN ⟨*nume entitate*⟩ PAR ⟨*criteriu*⟩; [⟨*calificare*⟩]
— dacă X_i punctează pe o realizare care aparține dicționarului se realizează acces la realizarea următoare conform ordinei în dicționar;
— dacă X_i=U se punctează pe prima realizare care aparține dicționarului;
— dacă X_i punctează pe o realizare care nu aparține dicționarului atunci este pus la nedefinit.

Exemplu:

M X1 = SUIVANT DE UN PERSOANA PAR NUME;

X1 va desemna adresa următoarei realizări PERSOANA al cărei nume este clasat în dicționar imediat după numele persoanei pe care X1 o puncta inițial.

Expresii aritmetice

Rol: definirea și utilizarea expresiilor de calcul aritmetic;
Sintaxa:
M Y_i = ⟨*expresie*⟩
⟨*expresie*⟩ : := ⟨*operand*⟩⟨*operator*⟩⟨*operand*⟩/
⟨*operand*⟩⟨*operator*⟩⟨*subexpresie*⟩/
⟨*expresie*⟩⟨*operator*⟩⟨*expresie*⟩
⟨*subexpresie*⟩ : := (⟨*expresie*⟩)
⟨*operand*⟩ : := ⟨*număr*⟩/Y_i
$\{\langle număr \rangle, Y_i\} \in [(-2^{31}+1), (+2^{31}-1)], \ i=\overline{1,25}$
⟨*operator*⟩ : :=/+/−/*///
⟨*operator*⟩ : trebuie încadrat între separatorii de limbaj (spațiu sau parantezele de descriere a subexpresiilor).

Prioritatea de efectuare a operațiilor este cea naturală iar ordinea de efectuare a operațiilor este:
1 — la priorități egale de la stînga la dreapta;
2 — la grupa pe subexpresii (prin paranteze); se evaluează mai întîi subexpresiile cele mai imbricate, după regula 1, după care evaluarea se efectuează ținînd cont de regula 2 și regula 1.

Tabelul **4.3. Prioritatea operațiilor**

operator	semnificație	prioritate operație
+	adunare	1
—	scădere	1
*	inmulțire	2
/	impărțire	2

Deoarece procesul de evaluare tratează de fapt *expresii de forma* ⟨*operand*$_1$⟩ ⟨*operator*⟩⟨*operand*$_2$⟩ iar operanzii implicați pot fi valori imediate sau variabile Yi, care pot fi poziționate la valoarea U (nedefinit), rezultatul evaluării unei astfel de expresii se prezintă ca în tabela 4.4.

Tabelul 4.4. **Rezultatul evaluării unei expresii binare**

⟨Operator⟩	Valoare ⟨operand$_1$⟩	Valoare ⟨operand$_2$⟩	Rezultat evaluare
adunare „+"	V_1	V_2	$V_1 + V_2$
	V_1	U	V_1
	U	V_2	V_2
	U	U	U
scădere „−"	V_1	V_1	$V_1 - V_2$
	V_1	U	V_1
	U	V_1	U
	U	U	U
înmulţire „*"	V_1	V_2	$V_1 \times V_2$
	V_1	U	U
	U	V_2	U
	U	U	U
împărţire „/"	V_1	V_2	$V_1 : V_2$
	V_2	U	U
	U	V_1	U
	U	U	U

Exemple :

```
M Y1 = Y1 + (Y2 – Y3) * Y1 + 10
M Y1 = Y1 * Y1
M Y2 = (Y1 * Y1)/(Y3 * Y2)
M Y3 = 100/Y1 /* aflarea restului */
M Y3 = 100 – Y3 * Y1 /* împărţirii a două numere */
```

Conversii

Rol : efectuarea conversiilor ⟨*zecimal despachetat*⟩-⟨*binar*⟩ sau ⟨*binar*⟩-⟨*zecimal despachetat*⟩ combinate eventual cu operaţiile de concatenare şi/sau extragere subşiruri de caractere.

1) Conversia ⟨*zecimal despachetat*⟩-⟨*binar*⟩

Sintaxa :

M $Y_i = Z_k$ unde : $i=\overline{1,25}$ şi $k=\overline{1,7}$

Specificaţii de utilizare :

– valoarea, reprezentată în zecimal despachetat, conţinută în variabila Z_k, este convertită în binar şi atribuită lui Y_i ;

– şirul de cifre zecimale, din Z_k, poate fi precedat de semnul „+" (număr pozitiv) sau „–" (număr negativ) ; în lipsa semnului numerele sînt considerate pozitive ;

– eventualele spaţii care preced sau urmează semnului sînt eliminate (numerele din Z_k pot fi aliniate sau la stînga sau la dreapta) ;

– dacă valoarea conţinută în $Z_k > (2^{31} - 1)$ sau $Z_k < (-2^{31} + 1)$ variabila Y_i ia valoarea U ;

– dacă valoarea conţinută în Z_k este numerică (spaţii între cifre sau caractere nume-(merice atunci lui Y_i i se atribuie valoarea U.

Acest tip de conversie poate fi combinat, eventual, cu extragere de subșiruri de caractere, astfel:
M $Y_i = Z_k$ [(] ⟨*poziție*⟩ [⟨*lungime*] [)]
unde:
(⟨*poziție*⟩ + ⟨*lungime*⟩) $\leqslant$ 30 și {⟨*poziție*⟩ ⟨*lungime*⟩} $\in$ [1,30]
⟨*poziție*⟩ : numărul caracterului de la care începe operația de conversație;
⟨*lungime*⟩ : lungimea, în caractere, șirului de tratat.

Exemple:

— Z_1 conține valoarea —123 (hexa: X'C0F1F2F3'),
rezultatul următoarelor operații este:
M Y1 = Z1 I Y1 ⇒ Y1 : —123;
M Y1 = Z1 1 2 I Y1 ⇒ Y1 : —1;
M Y1 = Z1 (2 3) I Y1 ⇒ Y1 : 123;
— Z1 = 'A121243 + 31' iar rezultatul următoarelor operații este:
M Y1 = Z_1 Y1 ⇒ Y1 : nedefinit;
M Y1 = Z1 (2 6) I Y1 ⇒ Y1 : 121243;
M Y1 = Z1 (8 3) I Y1 ⇒ Y1 : 31.

2) Conversia ⟨*binar*⟩-⟨*zecimal despachetat*⟩
Sintaxa:

M $Z_k = Y_i$, unde: $i = \overline{1,25}$ și $k = \overline{1,7}$

Specificații de utilizare:

— variabilei Z_k i se atribuie șirul de caractere reprezentate în zecimal despachetat care corespund valorii binare din variabila Y_i; atribuirea se face de la stînga la dreapta;
— dacă șirul de caractere rezultat depășește valoarea 30 (această situație poate apare la concatenări) va fi trunchiat din poziția corespunzătoare celui de al 30-lea caracter al lui Z_k;
— dacă Y_i conține o valoare pozitivă primul caracter din Z_k va fi spațiu iar dacă valoarea este negativă va fi „—";
— dacă Y_i conține valoarea U atunci lui Z_k i se atribuie valoarea „spații" (X'40').

Acest tip de conversie poate fi combinat, eventual, cu operația de concatenare șiruri de caractere, astfel:
M Z_k [(] ⟨*poziție*⟩ [)]$=Y_i$
⟨*poziție*⟩ $\in$ [1,30], indică caracterul din Z_k începînd cu care se depune rezultatul conversiei. Dacă lungimea șirului rezultat din conversie + ⟨*poziție*⟩ depășește valoarea 30 atunci apare situația de trunchiere. Dacă această sumă este mai mică de 30 caractere atunci după ultima cifră semnificativă se va completa, variabila Z_k, cu spații. Caracterul transferat în poziție este reprezentat de semnul numărului (spațiu sau „—").

Exemple:

M Y1 = 123
M Z1 = Y1 ; Z1 ⇒ Z1 : 123 (X'40F1F2F340...')
M Y1 = —Y1 I Y1 ⇒ Y1 :—123
M Z1(4) = Y1 I Z1 ⇒ Z1 : 123 — 123 (X'40F1F2F3COF1F2F3...').

Manipularea şirurilor de caractere

Operaţiile definite pe şiruri de caractere alfanumerice sînt:

— *extragerea* unui (sub)şir dintr-un (sub)şir alfanumeric şi afectarea lui unei variabile Z_i, eventual dintr-o poziţie bine determinată;

— *concatenarea* unui (sub)şir la un (sub)şir conţinut de o variabilă Z_i pornind, eventual, de la o poziţie bine determinată.

Concatenarea se efectuează de la stînga la dreapta prin eventuala completare cu spaţii a variabilei receptoare sau trunchiere a (sub)şirului emiţător.

Sintaxe:

1) M Z_i [[(] ⟨*poziţie*$_1$⟩ [)]] = '⟨*constantă alfanumerică*⟩' / Yk / Z_i / ⟨*citaţie caracteristică alfanumerică*⟩

⟨*poziţie*$_1$⟩ ∈ [1,30], $i = \overline{1,7}$ şi $k = \overline{1,25}$

⟨*citaţie caracteristică alfanumerică*⟩: reprezintă citaţia unei caracteristici de tip ⟨*cuvînt*⟩ ⟨*listă-de-valori*⟩ sau ⟨*cuvînt formal*⟩;

— dacă ⟨*poziţie*$_1$⟩ este absent atunci operaţia este operaţie de atribuire, altfel este operaţie de concatenare;

— ⟨*poziţie*$_1$⟩ indică poziţia de început a concatenerării (în variabila Z_i) conţinutului părţii drepte a cererii;

— dacă lungimea şirului de caractere de transferat din membrul drept este mai mare decît (30 – ⟨*poziţie*$_1$⟩) atunci membrul drept va fi trunchiat la dreapta;

— variabila Z_i, din membrul stîng, va fi completată, eventual, cu spaţii între poziţia ultimului caracter semnificativ conţinut (diferit de spaţiu sau ⊔) şi poziţia de început a concatenării.

2) M Z_i [[(] ⟨*poziţie*$_1$⟩ [)]] = Z_j [[(] ⟨*poziţie*$_2$⟩ [⟨*lungime*⟩] [)]]

{⟨*poziţie*$_2$⟩, ⟨*lungime*⟩} ∈ [1,30] şi $i,j = \overline{1,7}$

Specificaţii de utilizare:

— sînt valabile specificaţiile formei 1);

— ⟨*poziţie*$_2$⟩ indică indicele primului caracter al (sub)şirului de extras (primul caracter din şir are indicele 1);

— ⟨*lungime*⟩: indică lungimea în caractere a (sub)şirului de extras (în lipsă se ia în considerare toată lungimea zonei emiţătoare cu eventuala trunchiere a emisiei la receptor);

— dacă nu se doreşte trunchierea atunci trebuie respectată condiţia:
(30 – ⟨*poziţie*$_1$⟩) ⩾ (⟨*poziţie*$_2$⟩ + ⟨*lungime*⟩).

Exemple:

M Z2 = '1 2' M Z3 = '123456789'
M Z1 1 = Z2 I Z1 ⇒ Z1 : 12
M Z1 5 = Z3 (7 2) I Z1 ⇒ Z1 : 12⊔⊔78
M Z1 (8) = Z1 (1 7) I Z1 ⇒ Z1 : 12⊔⊔78⊔⊔12⊔⊔78

Observaţie:

Atribuirea M $Z_i = Z_i$ duce la eliminarea eventualelor spaţii care preced şi/sau urmează şirului de caractere semnificative conţinute de variabila Z_i.

Operația de extragere/concatenare, combinată eventual cu conversii, permite obținerea algoritmică, a unor valori cu semnificație particulară pentru utilizator, valori care pot fi atribuite caracteristicilor definite în baza sa de date (valori numerice atribuite unor caracteristici de tip cuvînt, construirea de chei compuse numerice cu putere de reprezentare mai mare decît $2^{31}-1$ etc.).

Dacă (sub)șirul conținut de membrul drept este nedefinit atunci variabila din membrul sting va fi pusă la nedefinit.

Dacă utilizatorul nu dorește acest lucru atunci este necesară testarea membrului drept prin comparare cu valoarea U și efectuarea condițională a atribuirii.

Rezervarea și formatarea bufferelor de lucru

SGBD-SOCRATE pune la dispoziția utilizatorilor săi posibilitatea structurării și rezervării unei (unor) zone de memorie (buffer) conform necesităților sale. Această structurare se realizează prin placarea pe zona de memorie a unui :
1) — formal logic, definit anterior placării ;
2) — format logic de entitate.

Zonele de memorie accesibile utilizatorului sînt identificate de numărul lor (0÷5).

Prin această placare utilizatorul definește de fapt structura zonei de memorie, adresele de intrare în această zonă (adresarea este realizată prin citirea identificatorilor caracteristicilor care compun structura logică) și tipul zonelor definite.

După această placare bufferul structurat va fi utilizat ca organ de intrare/ieșire pentru cererile aplicației SOCRATE care îl utilizează.

Regulile de utilizare a bufferelor depind de modul de prelucrare ales și numărul bufferului utilizat astfel :

a) în prelucrarea conversațională :

— fiecărui terminal îi este asociat, implicit, bufferul cu numărul 0 (zero) cu lungimea de 79 caractere. Orice tentativă de placare, pe acest buffer, a unei structuri logice cu lungimea mai mare provoacă o eroare sistem ;

— bufferele cu numerele 1÷5 pot fi rezervate, cu orice lungime $\leqslant$ 32ko, la cererea explicită a fiecărui utilizator. Aceste buffere se alocă în zona statică a partiției (parametrul DS al comenzii. OPTION a cărei lungime trebuie să acopere suma lungimilor tuturor bufferelor utilizate simultan. Eliberarea acestor buffere nu se efectuează în mod automat la cererea de deconexiune a unui utilizator (LOGOUT) ci numai prin citații specifice efectuate în program (LIBERER) ;

b) în „batch processing" : toate bufferele utilizate (0÷5) sînt rezervate în zona statistică. Eliberarea bufferelor se poate efectua explicit prin comanda LIBERER sau automat la închiderea sesiunii de lucru (LOGOUT) ;

c) cu interfața cu limbaje evoluate : singurul buffer utilizabil este cel cu numărul 0 (zero) a cărui alocare (mărime) se efectuează în programul scris în limbajul evoluat respectiv (prin acest buffer programul poate comunica cu SOCRATE dar nu există nici-o restricție asupra definirii unor buffere proprii programului respectiv).

Un buffer formatat conform structurii definite de un ⟨*formal logic*⟩ poate fi utilizat pentru :

— citirea unei înregistrări logice (de tip „COBOL" de exemplu) de pe o bandă magnetică, disc magnetic, cartelă perforată sau terminal ;

— a reprezenta un organ de intrare/ieșire a datelor pentru cererile SOCRATE ;

— eventuala sa scriere pe o bandă sau disc magnetic ;

— eliberarea sa.

Un buffer formatat conform formatului de descriere a unei realizări de entitate poate fi utilizat pentru :

— citirea unei realizări de entitate din baza de date ;

— a reprezenta un organ de intrare/ieșire a datelor, pentru cererile SOCRATE, diferit de baza de date dar interpretat conform structurii interne a acesteia ;

— eventuala scriere a sa, peste o realizare de entitate, în baza de date ;

— eliberarea sa.

Sintaxa de rezervare și structurare a unui buffer :

1) *Formal logic* :

M X_i = FORMAL ⟨*identificator*⟩ [{⟨*întreg*⟩ / Y_i}] [⟨*calificare*⟩]
DANS ⟨*identificator buffer*⟩ ⟨*număr buffer*⟩

{⟨*întreg*⟩ / Y_i} : valori pozitive întregi care desemnează o realizare a unui ⟨*formal logic repetitiv*⟩. Placarea unui ⟨*formal logic repetitiv*⟩ pe un buffer presupune placarea prealabilă a tuturor ascendenților săi (pînă se ajunge la ⟨*formal logic*⟩) pe același buffer.

[⟨*calificare*⟩] : se utilizează în cadrul ⟨*formatului logic repetitiv*⟩ și permite stabilirea apartenenței acestuia la o anumită structură de un nivel ierarhic superior.

⟨*calificare*⟩ ::= DE ⟨*identificator*⟩ [{⟨*întreg*⟩ / Y_i}][⟨*calificare*⟩]
[DE X_i]

⟨*identificator buffer*⟩ : un identificator ales de utilizator pentru a identifica bufferul în sensul lizibilității utilizărilor sale (ex. BUF, BUFFER, ZONA etc.).

⟨*număr buffer*⟩ ∈ [0,5] (numărul bufferului rezervat).

Variabila X_i, care desemnează bufferul, indică o structură de formal logic și adresa, în memoria centrală, a zonei, pe care se lucrează, structurată conform celor definite în acest format desemnat de ⟨*identificator*⟩.

Caracteristicile definite într-un formal logic, desemnat de o variabilă X_i, pot fi numai obiectul acțiunii comenzii M, astfel :

— M {⟨*cuvînt*⟩ / ⟨*listă-de-valori*⟩} / Z_i = ⟨*cuvînt formal*⟩ DE X_i

— M {⟨*valoare numerică*⟩ / Y_i} = {⟨*zecimal despachetat*⟩ / ⟨*zecimal împachetat*⟩}
DE X_i

— M ⟨*cuvînt formal*⟩ DE X_i = U/Z_i /⟨*cuvînt*⟩ / ⟨*lista-de-valori*⟩

— M {⟨*zecimal despachetat*⟩ / ⟨*zecimal împachetat*⟩} DE X_i = Y_i / U /
⟨*valoare numerică*⟩

2) *Format entitate* :

M X_i = FORMAT UN ⟨*nume entitate*⟩ [⟨*calificare*⟩] DANS
⟨*identificator buffer*⟩ ⟨*număr buffer*⟩

Caracteristicile definite în ⟨*nume entitate*⟩ devin disponibile conform tipului și structurii asociate, în zona identificată de ⟨*număr buffer*⟩. Aceste caracteristici se califică prin X_i și pot fi obiectul acțiunii comenzilor acceptate de tipul fiecăruia.

La momentul dorit de utilizator, pe acest buffer, se pot placa valorile asociate caracteristicilor definite ale unei realizări particulare a entității identificate de ⟨*nume entitate*⟩ astfel :

M X_j = ⟨*desemnare realizare de entitate*⟩
LIRE X_j DANS X_i ;

După efectuarea prelucrărilor dorite pe X_i utilizatorul poate să scrie acest buffer pe o realizare de entitate:

ECRIRE X_i DANS X_j

Mai mult acest buffer poate fi scris ca o înregistrare oarecare pe un fişier magnetic de unde poate fi citit şi rescris în baza de date. Evitarea tratării elementelor componente ale formatului de entitate în contextul respectiv poate fi realizată nedefinind bufferul prin placarea unor caracteristici de tip ⟨*formal logic*⟩, structurate în mod corespunzător, pe acelaşi buffer şi utilizarea elementelor definite în cadrul acestora. Formatarea bufferului conform unui ⟨*formal logic*⟩ poate să preceadă sau să urmeze (în acest caz lungimea caracteristicii) ⟨formal logic⟩ trebuie să fie cel mult egală cu cea a entităţii) structurării sale în format entitate.

Această formatare a bufferelor de memorie conform descrierii unui format de entitate dă posibilitatea unor intervenţii „manuale" în reprezentarea internă a caracteristicilor, ceea ce poate duce la consecinţe foarte grave privind consistenţa şi corectitudinea datelor.

Utilizarea acestei comenzi este posibilă din V.1.4 SOCRATE dar nu a fost făcută publică din raţiuni de păstrare a integrităţii şi securităţii bazelor de date. Practic, prin programele sale scrise în LMD, utilizatorul nu poate aduce, în mod necontrolat o legătură de tip ⟨*inel*⟩ — ⟨*referire*⟩ la incoerenţă fizică, lucru care devine posibil cu utilizarea acestei comenzi. Sintaxa şi modul de utilizare au fost prezentate deoarece ITCI în V.1.6.R o anunţă ca o noutate a versiunii, de aceea propunem titularilor de software sursă blocarea formei de scriere în baza de date, lăsînd disponibilă numai forma de citire.

Şi numai cu această formă utilizatorul poate economisi foarte mult timp de lucru în sensul că transferul punctual în/din buffer/ baza de date se poate efectua fără angrenarea proceselor de conversie din/în formatul de reprezentare internă propriu lui SOCRATE în/din formatul de reprezentare propriu caracteristicilor din ⟨*formal logic*⟩.

Specificaţii privind desemnarea şi utilizarea bufferelor:

a) Un buffer poate fi desemnat de mai multe variabile X_i şi pote fi placat cu mai multe caracteristici formal:

M X1 = FORMAL ⟨$form_1$⟩ DANS BUF 1
M X2 = FORMAL ⟨$form_1$] DANS BUF 1
M X3 = FORMAL ⟨$form_2$⟩ DANS BUF 1

— X1 şi X2 desemnează acelaşi buffer pentru acelaşi formal;

— dacă lungime ⟨$form_2$⟩ este mai mică sau egală cu lungime ⟨$form_1$⟩ variabilele X1 X2 şi X3 vor desemna acelaşi buffer (echivalează cu o redefinire a bufferului);

— dacă lungime ⟨$form_2$⟩ este mai mare decît lungime ⟨$form_1$⟩ atunci X3 va desemna un nou buffer cu numărul 1, izolat în întregime de zona definită de X1 şi X2. O citire în bufferul 1 va încărca această zonă accesibilă în mod unic prin X3. Variabilele X1 şi X2 vor desemna prima zonă rezervată; X1 şi X2 pot fi redefinite în bufferul adresat de X3 prin cererile:

M X1 = FORMAL ⟨$form_1$⟩ DANS BUF 1
M X2 = FORMAL ⟨$form_1$⟩ DANS BUF 2

Acest mod de lucru permite de fapt o asociere dinamică la momentul execuţiei programului de cereri (se testează, de exemplu, valoarea unui element, după citire, şi în funcţie de condiţiile impuse acesteia se poate redefini asocierea).

b) Funcțiile de citire/scriere se efectuează cu ajutorul comenzilor LIRE/ECRIRE care trebuie să citeze numărul bufferului asociat variabilei X_i;

c) ștergerea unui buffer (pe care s-a placat un format logic) asociat unei variabile X_i se realizează cu comanda S X_i (nu provoacă „suprimarea" ci suprascriere cu spații);

d) eliberarea unui buffer în cadrul aceleiași sesiuni de lucru se realizează explicit prin comanda:

LIBERER ⟨*identificator buffer*⟩ ⟨*număr buffer*⟩

Exemplu:

Cererea: M X1 = FORMAL MACHETA DANS BUF 1

provoacă *rezervarea*, în memoria internă, *unei zone* care va fi *desemnată*, la nivelul limbajului de cereri, prin X1.

Mărimea bufferului rezervat este aceea a formatului logic MACHETA, definit în structura bazei de date (80 caractere).

Utilizarea variabilei X1 se face în aceeași manieră cu utilizarea unui X_i definit ca adjectiv de desemnare.

Eventualele cereri de intrare/ieșire trebuie să utilizeze numărul 1 ca număr al bufferului:

LIRE DANS BUF 1
ECRIRE BUF 1 DANS BANDE 01

Identificatorul bufferului care urmează cuvîntului DANS este utilizat numai pentru clarificarea sensului frazei (nu este controlat de sistem).

Variabilele X_i reprezintă singurii calificatori posibili ai identificatorilor interni unui format logic.

Cererea M X2 = FORMAL CALIFICATIVE 3 DE X1

afectează variabilei X2 adresa, în memorie, a celui de-al 3-lea CALIFICATIV din formatul logic MACHETA. La modul general o cerere de acest tip permite afectarea la o variabilă X_i a adresei unui format logic, eventual repetitiv, intern unui alt format logic deja asociat unui buffer.

Comanda I (editare)

Rol: interogarea caracteristicilor care constituie structura bazei de date și/sau a variabilelor de lucru X_i, Y_i și Z_i.

Efectul aplicării comenzii asupra obiectelor acțiunii este concretizat în editarea conținutului acestora. Editarea se poate realiza în funcție de opțiunile utilizatorului, în două forme:

1 — editare standard (fără punere în formă);
2 — editare cu format (cu punere în formă).

Alegerea uneia sau celeilalte forme de editare este la latitudinea utilizatorului și se realizează prin alegerea variantei sintactice a comenzii I.

Editare standard

Acest tip de editare este propriu SGBD-SOCRATE și se realizează prin punerea automată în pagină a obiectelor acțiunii după un format și într-o formă specifică fiecărui tip de obiect implicat.

Sintaxa :

I ⟨*identificator*⟩ [⟨*calificare*⟩] / $X_i/Z_i/Y_i$ '*constantă alfanumerică*'

Specificații de utilizare :

Rezultatul execuției comenzii depinde (în sensul formei și formatului de editare) de tipul obiectelor implicate astfel :

— înaintea unui identificator al unei caracteristici de tip ⟨*cuvînt*⟩ ⟨*valoare numerică*⟩, ⟨*listă-de-valori*⟩ sau variabilă de lucru (X_i, Y_i, Z_i) imprimarea se realizează sub forma :

{⟨*identificator*⟩ / $X_i/Y_i/Z_i$} : *valoare*

Dacă ⟨*identificator*⟩ sau X_i desemnează o caracteristică de tip ⟨*referire*⟩ valoarea imprimată reprezintă numărul de realizare al entității referite cu o eventuală calificare numerotată.

Dacă valoarea atribuită caracteristicii sau variabilei este nedefinit atunci se vor imprima spații :

— înaintea unui identificator al unei caracteristici de tip ⟨*entitate*⟩, ⟨*inel*⟩ ⟨*invers*⟩, ⟨*bloc*⟩ sau X_i desemnînd o realizare de entitate se editează toate caracteristicile entității sau blocului sau toate caracteristicile entităților înlănțuite sau inversate cu imprimarea numelor și valorilor atribuite tuturor caracteristicilor incluse structural. Pentru o aranjare sugestivă în pagină se folosește tehnica identării caracteristicilor de nivel superior. Dacă o entitate conține o caracteristică de tip ⟨*inel*⟩ sau ⟨*invers*⟩ se imprimă numele entităților înlănțuite sau inversate indicîndu-se numărul lor de realizare și eventuala calificare fără imprimarea caracteristicilor interne și a valorilor asociate acestora.

În concluzie, utilizarea comenzii I în fața unui ⟨*identificator*⟩ de tip ⟨*entitate*⟩, ⟨*bloc*⟩, ⟨*inel*⟩ sau ⟨*invers*⟩ se comportă ca o punere în factor a acestei comenzi pentru toate caracteristicile incluse structural

— înaintea unei caracteristici de tip ⟨*text*⟩ :

⟨*identificator*⟩ : TEXT :

linie 1
linie 2
. } valorile atribuite liniilor generate în caracteristică
.
.

— comanda I '⟨*constantă alfanumerică*⟩' provoacă imprimarea constantei astfel :

⟨*constantă alfanumerică*⟩

Dacă șirul de caractere care definesc constanta conține mai mult de 30 caractere atunci acest lucru va fi semnalat la compilarea programului, printr-un mesaj de atenționare, iar la execuție se va realiza trunchierea la 30 de caractere.

Exemple :

— **imprimă numele, prenumele și data nașterii al primei persoane definite în baza de date :**

```
POUR UN PERSOANA
  I NUME I PRENUME I DATA-NASTERII FIN ?
NUME : POPESCU          /* caracteristica de tip cuvînt */
PRENUME : ION
DATA-NASTERII           /* imprimarea unui bloc */
  AN : 1913             /* caracteristici de tip valoare numerică */
  LUNA : 1
  ZIUA : 13
```

```
PERS-LOCN : REF          /* caracteristică de tip    */
   LOCALITATI 100        /* referire                 */
```

— imprimarea tuturor realizărilor entităţii NATIONALITATE-CETATENIE

```
I TOUT NATIONALITATE-CETATENIE ?
NATIONALITATE-CETATENIE 1
   COD : 1
   DEN : ROMANI
NATIONALITATE-CETATENIE 2
   .
   .
   .
```

Editare cu format

Acest tip de editare permite utilizatorului să aranjeze în pagină valorile asociate obiectelor acţiunii comenzii conform necesităţilor sale.

Sintaxa :

I (⟨A⟩ [⟨B⟩]){ *⟨identificator⟩* [*⟨calificare⟩*] /Y_i/Z_i/'*⟨constantă alfanumerică⟩*'}

⟨A⟩ ::= *⟨număr⟩* /Y_i *pozitiv*/+*⟨număr⟩* /+ Y_i *pozitiv*

⟨B⟩ ::= *⟨număr⟩* /Y_i *pozitiv*/– *⟨număr⟩* /Y_i *negativ*

⟨număr⟩ : număr pozitiv aparţinînd intervalului [0,120]

Specificaţii de utilizare :

Comanda permite completarea unui buffer de editare (rezervat implicit de sistem) cu valorile obiectelor acţiunii comenzii specificate.

Completarea bufferului se realizează conform unei poziţii de început specificare (⟨A⟩), pozitive faţă de care se specifică, eventual, lungimea de editare şi modul de aliniere la poziţie (⟨B⟩).

Dacă ⟨B⟩ este absent atunci lungimea transferului şi a zonei rezervate este dată de lungimea, în caractere de editare, a obiectului acţiunii comenzii.

Semnul parametrului ⟨B⟩ determină modul de aliniere a valorii obiectului acţiunii, comenzii, desemnat, faţă de poziţia indicată în ⟨A⟩, astfel :

– dacă ⟨B⟩ este pozitiv completarea se efectuează de la stînga la dreapta (aliniere stînga) ;

– dacă ⟨B⟩ este negativ completarea se efectuează de la dreapta spre stînga (aliniere dreapta).

Valoarea absolută a lui ⟨B⟩ reprezintă lungimea pe care se imprimă (deci transferă în buffer) valoarea conţinută de obiectul acţiunii comenzii. Eventualele diferenţe de lungime între valoarea lui ⟨B⟩ şi lungimea valorii de imprimat se rezolvă astfel :

– ⟨B⟩ mai mare decît lungimea valorii : se transferă valoarea iar restul caracterelor sînt făcute spaţiu ;

– ⟨B⟩ mai mic decît lungimea valorii : are loc o trunchiere stîngă sau dreaptă a valorii în funcţie de semnul lui ⟨B⟩ (negativ respectiv pozitiv).

Valoarea atribuită parametrului ⟨A⟩ reprezintă :

– {*⟨număr⟩* /Y_i *pozitiv*} : coloana de la care începe zona destinată recepţionării pentru imprimare a valorii obiectului acţiunii comenzii ;

– {+ *număr* / + Y_i *pozitiv*} : coloana de la care începe zona destinată recepţionării valorii obiectului acţiunii comenzii este dată de poziţia în care s-a ajuns prin precedenta comandă I formatat + *⟨număr⟩* sau + Y_i. Conţinutul zonei dintre poziţia în care s-a ajuns pe lungime + *⟨număr⟩* sau Y_i este lăsat nemodificat.

Dacă ⟨*număr*⟩ sau Y_i este 0 (zero) atunci comanda realizează concatenarea valorii obiectului acţiunii comenzii, concatenare începînd cu poziţia la care s-a ajuns cu precedenta comandă I formatat în buffer.

Scrierea bufferului pe suportul de ieşire (imprimantă sau terminal) se realizează cu un **ordin ECRIRE**. Execuţia unui ordin ECRIRE păstrează bufferul nemodificat, astfel încît ⟨*n*⟩ ordine ECRIRE succesive (fără I standard între ele) provoacă imprimarea de ⟨*n*⟩ ori a conţinutului bufferului.

Prima comandă I formatat, din programul utilizator, provoacă anularea bufferului de ieşire. Această anulare este realizată şi în momentul detectării oricărei comenzi I standard (de exemplu I ''·reprezintă imprimarea unei interlinii).

Asupra modului de funcţionare a comenzii se fac precizările :

– numele obiectelor acţiunii comenzii nu este imprimat ;
– valoarea nedefinit este reprezentată cu spaţii ;
– numerotarea coloanelor din buffer începe cu poziţia 1.

Observaţii :

— deoarece SGBD-SOCRATE este construit în vederea optimizării consumului de resurse, valorile nedefinite sînt reprezentate prin spaţii şi, aceste spaţii sînt reacoperite în cazul scrierii cu ⟨A⟩ de forma + ⟨*număr*⟩ *I* + Y_i. În acest context dacă utilizatorul doreşte o aliniere exactă (în coloane precise) a valorilor editate trebuie să folosească pentru ⟨A⟩ forma ⟨*număr*⟩ sau Y_i pozitiv ;
— faptul că bufferul de ieşire nu este anulat la fiecare comandă I formatat este determinat de oferirea posibilităţii menţinerii în factor a unei (unor) valori stabilite de utilizator pentru mai multe linii de afişare consecutive ;
— în cazul în care valoarea lui Y_i utilizat pentru a desemna parametrul ⟨A⟩ este negativă sau valoarea atribuită lui Y_j pentru a desemna parametrul ⟨B⟩ depăşeşte graniţele intervalului se va abandona prelucrarea sesiunii în curs, poziţionînd codul de retur al fazei la o valoare mai mare decît 127. Această proprietate poate fi utilizată cu succes de utilizator pentru a întrerupe forţat prelucrările sale cu marcarea codului de retur al fazei (valoarea codului de retur poate decide prelucrările ulterioare de executat utilizînd de exemplu, proceduri catalogate în care se testează codul de retur şi se determină condiţional înlănţuirea fazelor de executat) ;
— pentru valorile numerice sau variabile Y_i se va folosi forma —⟨*număr*⟩ sau $-Y_j$ pentru parametrul ⟨B⟩ (permite interpretarea corectă a semnalului valorii sau variabilei).

Exemple :

— editarea registrului numerelor matricole al persoanelor din baza de date sub forma :

REGISTRUL NUMERELOR MATRICOLE PAGINA : * * * * * *

NR. CRT.	MATRICOL	NUME ŞI PRENUME	DATA NAŞTERII			OBSERVAŢII
9(5)	X(13)	x(60)	9(4)	99	99	X(23)

Editarea capului de tabel se va face la fiecare pagină nouă.
⟨*apel procesor LMD*⟩
M Y3 = 0 /* contor număr curent
M Y1 = 1 /* contor pagini editate
M Y2 = 0 /* contor număr rînduri pe pagină

```
/* parcurgerea realizărilor entităţii persoana se face în ordinea
POUR TOUT PERSOANA X1 PAR COD; /* strict crescătoare a codurilor
 M Y3 = Y3 + 1 /* majorare număr curent la noua persoană disponibilă
 SI Y2 = 0 /* dacă numărul de rînduri scrise pe pagină este o
 ALORS /* atunci
  EXEC PAGESKIP /* program IMT realizînd salt la pagina nouă
  I (1) 'REGISTRUL NUMERELOR MATRICOLE'
  I (100) 'PAGINA':
  I (+5 –5) Y1 /* cadrare dreapta număr pagina
  ECRIRE /* scrie linia */
  I (1)  '*→ 30 ←*'
  I (+0)'*→ 30 ←*'  }
  I (+0)'*→ 30 ←*'  }  linia de 120 caractele '*'
  I (+0)'*→ 30 ←*'  }
  ECRIRE
  I (1)  '* NR. * MATRICOL' I (21) '* NUME I PRENUME'
  I (82) '*'           I (95) '* OBSERVAŢII' I (119) '*'
  ECRIRE
  I (1) '* CRT. *' I (21) '*' I (82 15)'* DATA NASTERII *'
  I (119) '*' ECRIRE
  I (1) '*→ 30 ←*'    I (+0) '*→ 30 ←*'
  I (+0) '*→ 30 ←*'  I (+0) '*→ 30 ←*'
  ECRIRE
  M Y2 = 5  /* număr linii ocupat de capul de tabel
  M Y1 = Y1 + 1 /* contorizare pagini editate
FIN
 I 1) '*'      I (6 – 5) Y3 I (7) '*'
 I (+0) COD   I (21) '*'
 I (+0) NUME I (+1) PRENUME I (82) '*'
 I (86 –4) AN DE DATA-NASTERII I (87) '*'
 I (98 –2) LUNA DE DATA-NASTERII I (90) '*'
 I (95 –2) ZI DE DATA-NASTERII I (96) '*' I (119) '*'
  ECRIRE
 M Y2 = Y2 + 1  /* contor număr linii
 SI Y2>95  /* număr maxim linii/pagină
 ALORS
  M Y2 = 0 /* dictează saltul la pagină nouă
            /* (90 de persoane/pagină)
 FIN
FIN ?
```

Comanda SI (dacă)

Rol : execuţia condiţională a unei suite de cereri (definirea cererilor condiţionale) ;

Sintaxa :	*Echivalent* :

SI ⟨condiţie⟩
ALORS
⟨suită de cereri₁⟩
SINON
⟨suită de cereri₂⟩
FIN
	IF (C)
	THEN /* condiţie evaluată A */
	g
	ELSE /* condiţie evaluată F */
	h
	ENDIF;

Reprezentarea simbolică a acestei structuri alternative este (fig. 4.1) :

⟨*suita de cereri*⟩ ::= ⟨*cerere*⟩/⟨*cerere*⟩⟨*suita de cerere*⟩

⟨*cerere*⟩ : orice tip de cerere exprimat conform specificaţiilor LMD-SOCRATE.

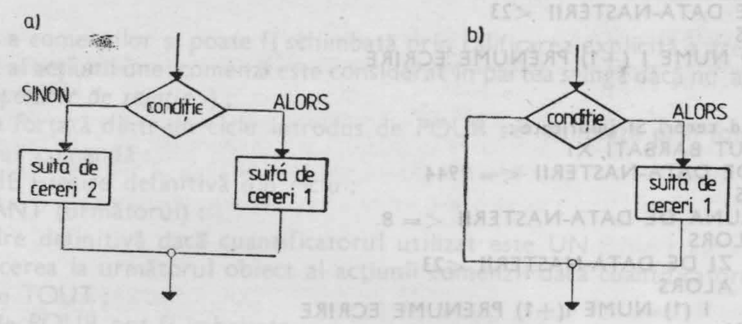


Fig. 4.1. Structuri alternative

Specificații de utilizare :

- clauza SINON poate lipsi ;
- fiecare SINON se asociază celui mai apropiat ALORS liber ;
- fiecare SI se închide cu un FIN ;
- fiecare SI se asociază celui mai apropiat FIN liber ;
- un ALORS sau FIN este liber atunci cînd nu este cerut de sintaxa nici-uneia din cererile incluse în ⟨suita de cereri⟩.

Observații :

1) Atunci cînd, în programul său, un utilizator parcurge realizările unui (sub)ansamblu selectate conform unui filtru iar expresia condițională a acestuia este complexă (cuprinde mai mult de două condiții) este preferabilă înlocuirea acestei expresii condiționale cu cereri de tip SI aplicate condițiilor simple, care compun expresia, stabilind prin imbricarea lor sau poziția secvențială a lor o ordine de excludere mutuală.

Această transformare duce la o economie de timp în sensul că valorile nu se mai supun evaluării integrale a expresiei condiționale (o expresie condițională se evaluează la A sau F) ci unor evaluări punctuale bazate pe o prioritate de excludere (dacă o condiție nu este îndeplinită nu se mai continuă procesul de testare).

Exemplu :

Se dorește listarea numelui și prenumelui tuturor bărbaților născuți înainte de 23 AUGUST 1944.

Această cerere de informare se poate exprima cu ajutorul unui filtru astfel :

```
POUR TOUT BARBATI X1 AYANT
    AN DE DATA-NASTERII <=1944
  ET LUNA DE DATA-NASTERII <=8
  ET ZI DE DATA NASTERII <23 ; DE X0
    I (1) NUME I (+1) PRENUME ECRIRE
FIN ?
```

Echivalentul acestei cereri exprimat prin condiții SI este :

```
POUR TOUT BARBATI X1 DE X0
  SI AN DE DATA-NASTERII >1944
    ALORS
      SUIVANT /* treci la analiza următoarei realizări */
  FIN
  SI LUNA DE DATA-NASTERII >8
    ALORS
      SUIVANT /* treci la analiza următoarei realizări */
  FIN
```

```
 SI ZI DE DATA-NASTERII <23
  ALORS
   I (1) NUME I (+1) PRENUME ECRIRE
  FIN
FIN ?
```

sau utilizînd cereri SI imbricate :

```
POUR TOUT BARBATI X1
 SI AN DE DATA-NASTERII <= 1944
  ALORS
   SI LUNA DE DATA-NASTERII <= 8
    ALORS
     SI ZI DE DATA-NASTERII <23
      ALORS
       I (1) NUME I(+1) PRENUME ECRIRE
      FIN
    FIN
  FIN
FIN ?
```

2) Dacă același obiect al acțiunii comenzii SI participă în exprimarea mai multor condiţii simple este preferabilă transferarea sa prealabilă aplicării expresiei într-o variabilă de lucru şi utilizarea acestei variabile la formarea expresiei.

3) Afirmaţiile privind descompunerea expresiilor condiţionale complexe (care utilizează obiecte diferite din baza de date) prin stabilirea unei ierarhii de excluderi mutuală sînt valabile şi pentru expresiile condiţionale utilizate la comanda SI.

4) Utilizatorul trebuie să ţină cont, la utilizarea comenzii SI, de modul de evaluare a expresiilor condiţionale iar la imbricarea comenzilor de tip SI este indicată utilizarea tehnicilor de simplificare a funcţiilor logice booleene.

Comanda POUR

Rol : gruparea unei suite de cereri ;

Sintaxa :

POUR {X0 / X_i/ ⟨*citaţie de caracteristică*⟩}
  ⟨*suita de cereri*⟩
FIN

unde : $i = \overline{1,9}$

X0 : desemnează blocul „FICHIER" (baza de date) ;

X_i : desemnează adresa unei realizări de entitate ;

⟨*citaţie de caracteristică*⟩ : citarea unei caracteristici de tip ⟨*bloc*⟩, ⟨*referire*⟩, ⟨*entitate*⟩, ⟨*inel*⟩ sau ⟨*invers*⟩.

Specificaţii de utilizare :

– în cazul în care ⟨*citaţie de caracteristică*⟩ se referă la o caracteristică de tip ⟨*entitate*⟩, ⟨*inel*⟩ sau ⟨*invers*⟩ această citaţie reprezintă o ⟨*desemnare de (sub)ansamblu*⟩ care implică modul de acces şi, eventual, selecţie a realizărilor ;

– o cerere POUR determină :

a) la citarea unei caracteristici de tip ⟨*bloc*⟩ şi ⟨*referire*⟩ sau la utilizarea cuantificatorului UN pentru ⟨*desemnare de (sub)ansamblu*⟩ o simplă punere în factor a calificatorului respectiv ;

b) la utilizarea cuantificatorului TOUT pentru ⟨*desemnare de (sub)ansamblu*⟩ o execuţie ciclică a ⟨suitei de cereri⟩ pentru fiecare realizare selectată a(sub)ansamblului cu punerea în factor a calificatorului respectiv ;

– fiecare POUR se asociază celui mai apropiat FIN liber ;

– valabilitatea contextuală a calificatorului pus în factor de POUR ţine pînă la FIN asociat (calificarea este implicită pentru toate caracteristicile aflate în partea

stîngă a comenzilor și poate fi schimbată prin calificarea explicită a acestora ; un obiect al acțiunii unei comenzi este considerat în partea stîngă dacă nu apare după un ⟨*operator de relație*⟩) ;

— ieșirea forțată dintr-un ciclu introdus de POUR se realizează prin folosirea cuvîntului comandă :
SORTIE : ieșire definitivă din ciclu ;
SUIVANT (următorul) :
1) ieșire definitivă dacă cuantificatorul utilizat este UN ;
2) trecerea la următorul obiect al acțiunii comenzii dacă cuantificatorul utilizat este TOUT ;
— cererile POUR pot fi imbricate ;
— dacă imbricarea se realizează pe ⟨*desemnare de (sub)ansamblu*⟩ atunci este indicată utilizarea ⟨*pronumelor de apel*⟩ X_i pentru desemnarea (sub)ansamblului care reprezintă obiectul acțiunii. Aceste ⟨*pronume de apel*⟩ pot fi utilizate la realizarea eventualelor rupturi de context necesare la execuție ;
— derularea unui ciclu POUR poate fi oprită, în conversațional, prin tastarea caracterului „atenție" (C1N) sau apăsare pe tasta BREAK). Un ciclu căruia i s-a oprit derularea poate fi relansat prin comanda RESTART.

Utilizarea cererilor POUR, imbricate, pe ⟨*desemnare de (sub)ansamblu*⟩ permite programatorului să „navigheze", conform necesităților sale, în datele stocate în baza sa de date prin parcurgerea lineară sau încrucișată a „căilor de parcurgere" definite. Prin utilizarea „rupturilor de context" are posibilitatea să se refere, la un nivel de îmbricare inferior, la orice obiect manipulat la un nivel superior și/sau să părăsească /adopte o cale de parcurgere în favoarea/defavoarea alteia.

Vom da, în continuare, un exemplu de parcurgere a unei structuri cu o foarte mare putere de reprezentare a ierarhiei datelor și anume o structură de entitate care conține declarația de ⟨*inel*⟩ și ⟨*referire cu inel*⟩ asociată. O astfel de structură poate reprezenta utilizînd termenii proprii teoriei structurilor arborescente, „frunze ramuri, arbori, păduri sau păduri de arbori". Încadrarea datelor în una din aceste structuri este dependentă de proprietățile intrinseci ale acestora reprezentate de modul de realizare a subordonării. Pentru exemplificare vom utiliza structura, descrisă în cap. 3, administrativ-teritorială a R.S.R. ; scopul programului este de a edita pentru localitățile definite la primul nivel localitățile aflate la un nivel ierarhic inferior. Evidențierea nivelului se va realiza prin decalarea, spre dreapta, cu un număr de 5 caractere a poziției de început a imprimării.

Exemplu :

```
⟨apel procesor LMD⟩
POUR TOUT LOCALITATI X1 /* localitățile de nivel 1 nu */
   AYANT PAS LOC-SUP; /* sînt subordonate            */
   I (5 — 5) COD I(+1) DEN1 I (+0) DEN ECRIRE
  POUR TOUT LOC-SUB X1 /* nivelul 2 */
   I (10 — 5) COD I (+1) DEN1 I(+0) DEN2 ECRIRE
    POUR TOUT LOC-SUB X1 nivelul 3 */
     I (15 — 5) COD I (+1) DEN1 I(+0) DEN2 ECRIRE
    POUR TOUT LOC-SUB X1 /* nivel 4 */
     I (20 — 5) COD I (+1) DEN1 I (+0) DEN2 ECRIRE
    POUR TOUT LOC-SUB X1 /* nivel 5 */
     I (25 — 5) I (+1) DEN1 I(+0) DEN2 ECRIRE
     SI EXISTE LOC-SUB
       ALORS
        I(30) 'ERR.FF.EROARE FATALA DE DATE ?'
```

```
            SORTIE
        FIN
FIN FIN FIN FIN FIN ?
```

Notă : localitățile existente la un nivel de subordonare nu sînt de același tip administrativ. Tipul poate fi identificat prin precizarea sa explicită în denumire sau prin introducerea unor niveluri vide care conțin numai legăturile pentru a le plasa la un nivel dependent de tip.

Comanda FAIRE

Rol : repetarea execuției unei ⟨suite de cereri⟩ de un număr nedeterminat de ori. Numărul de execuții, locul, momentul și condițiile de relansare sau abandonare a execuției trebuiesc să fie stabilite explicit de utilizator prin comenzi specifice.

Sintaxa :

FAIRE
⟨*suita de cereri*⟩
FIN

Specificații de utilizare :

— fiecărei comenzi FAIRE i se asociază cel mai apropiat FIN liber ;
— execuția unui ciclu FAIRE poate fi întreruptă, în conversațional, în orice moment prin tastarea caracterului „atenție" și relansată prin comanda RESTART (între momentul întreruperii și cel al relansării este permisă numai utilizarea comenzilor de primul nivel).

Pentru un ciclu de FAIRE se fac precizările :

— *relansarea* se realizează prin comanda REFAIRE ;
— *ieșirea* se realizează prin comanda SORTIE ;

Observații :

Numărul, locul și condițiile de execuție a comenzilor REFAIRE/SORTIE este stabilit de utilizator. Fiecare comanda REFAIRE/SORTIE se asociază nivelului introdus de comanda FAIRE in care este definită. Dacă unei comenzi FAIRE nu i se asociază o comandă REFAIRE atunci ⟨*suita de cereri*⟩ este executată o singură dată (integral dacă nu există comenzi SORTIE care să o fragmenteze) și permite numai definirea unui nivel inferior în program. Prin utilizarea comenzii SORTIE permite execuția condițională a porțiunilor din ⟨suita de cereri⟩ fără utilizarea regrupării acestuia sub o comandă *SI*.

Exemplul 1 :

— calculul factorialului unui număr oarecare ⟨N⟩ :

```
⟨apel procesor LMD⟩
M Y1 = 1 /* variabila pentru calculul factorialului */
M Y2 = ⟨N⟩
FAIRE /* lansare ciclu de execuție */
  SI Y2 = U OU Y2 ⟨ = 0 /* condiția de ieșire din ciclu */
    ALORS
      SORTIE /* ieșire din ciclu */
  FIN
  M Y1 = Y1 * Y2
  M Y2 = Y2 — 1
  REFAIRE /* relansează ciclul */
FIN
I Y1 ?
```

Exemplul 2 :

— căutarea unei (unor) persoane din baza de date după cod şi imprimarea numelui şi prenumelui acestora :

```
⟨apel procesor LMD⟩
D X1 /* anulare variabilă Xi pentru asigurare reentantă la nivel sesiune */
FAIRE I 'INTRODUCETI CODUL PERSOANEI'
  M Z1 = U Z1 = EXT
  SI Z1 = 'TERMINAT' /* condiţia de ieşire din ciclu */
    ALORS
      SORTIE
  FIN
  M X1 = UN PERSOANA AVEC COD = Z1 ;
  SI PAS X1 /* dacă persoana nu există în B.D. se dă */
    ALORS /* mesaj de eroare
      I (1) 'ERR.01.PERSOANA NU EXISTA ?'
      ECRIRE
      REFAIRE /* reluare ciclu în caz de eroare */
  FIN
  POUR X1 /* în caz de eroare această secvenţă nu se execută */
    I(16) NUME I (+1) PRENUME ECRIRE
  FIN
  REFAIRE /* reluare ciclu normală */
FIN D X1 /* anulare variabilă Xi pentru asigurarea reentranţei */
         /* programelor la apelul în cadrul aceleiaşi sesiuni */
? /* lansare faza compilare şi execuţie a programului */
```

Comanda DEPUIS

Rol : lansarea unui ciclu POUR care se desfăşoară asupra realizărilor unei caracteristici de tip ⟨*entitate*⟩ sau ⟨*invers*⟩, începînd cu o anumită realizare.

Sintaxa :

DEPUIS {⟨*număr*⟩ Y_i}
  POUR ⟨*citaţie ansamblu*⟩
    ⟨*suita de cereri*⟩
  FIN

{⟨*număr*⟩/Y_i} ∈ [0, *Nmre*] ∩ N

⟨*citaţie ansamblu*⟩ :: = TOUT {⟨*nume entitate*⟩/⟨*nume invers*⟩}[X_i]
[⟨*calificare*⟩]

Specificaţii de utilizare :

— parcurgerea realizărilor specificate în ⟨*citaţie ansamblu*⟩ se face în ordinea crescătoare a numerelor de realizare începînd cu prima realizare, care există, cu număr de realizare mai mare decît cel furnizat de utilizator prin ⟨*număr*⟩ sau Y_i ;
— realizarea cu numărul ⟨*număr*⟩ (sau Y_i) poate să nu existe ;
— deşi sintaxa definită pentru ⟨*citaţie ansamblu*⟩ arată acest lucru, precizăm că este interzisă parcurgerea cu ⟨*criteriu de sortare*⟩ sau ⟨*filtru*⟩ de selecţie.

Observaţie :

Această comandă permite, în cazul parcurgerii secvenţiale a unui ansamblu (după ⟨*şir de biţi*⟩), reluarea prelucrării din locul în care s-a întrerupt în urma unui incident hardware sau la comanda explicită a utilizatorului. Mai mult utilizatorul poate fracţiona prelucrările voluminoase prin construirea unor „puncte de reluare" a parcurgerii ansamblului sau poate exploata parţial ansamblul (între anumite limite de variaţie a numerelor de realizare).

Exemple :

– imprimarea localităţilor definite după localitatea cu codul 30465 :

```
M Y1 = NUMDE UN LOCALITATI AVEC COD = 30465 ;
DEPUIS Y1
  POUR TOUT LOCALITATI
    I (6 – 5) COD I (+1) DEN1 I (+0) DEN2 ECRIRE
FIN ?
```

– imprimarea localităţilor aparţinînd unui (unor) judeţe cu construirea unor „puncte de reluare" la fiecare 90 de localităţi listate.

- Prelucrarea se va efectua conform unui parametru furnizat de utilizator cu structura :

⟨prel⟩⟨ji⟩⟨jf⟩⟨nrloc⟩

⟨prel⟩ ::= T/C

T – termină prelucrarea ;

C – continuă prelucrarea cu acest parametru.

Dacă valoarea atribuită lui *⟨prel⟩* diferă de T sau C atunci se semnalează eroarea şi se solicită din nou parametrul.

⟨ji⟩ : două caractere numerice precizînd numărul judeţului de la care începe prelucrarea. Dacă acest parametru este 0 sau 1 prelucrarea va începe cu primul judeţ definit. Acest parametru este obligatoriu ;

⟨jf⟩ : două caractere numerice precizînd judeţul la care se va termina prelucrarea şi se va solicita un nou parametru.
Dacă *⟨jf⟩* este absent atunci vor fi prelucrate toate judeţele care urmează lui *⟨ji⟩*. Dacă valoarea atribuită lui *⟨jf⟩* este mai mică decît a lui *⟨ji⟩* se semnalează eroare şi se solicită din nou parametrul.

⟨nrloc⟨ : numărul localităţii (5 caractere), din cadrul primului judeţ *⟨JI⟩* de la care începe listarea. Dacă nu există acest parametru se listează toate localităţile aparţinînd judeţului. Pentru judeţele care urmează lui *⟨JI⟩* acest parametru este ignorat.

Exemplu :

```
⟨apel procesor LMD⟩
FAIRE
 I (1) 'INTRODUCERI PARAMETRII PRELUCRARII'
 I (+1) 'SUB FORMA ⟨PJIJFLLLLL⟩ UNDE : ' ECRIRE I ' '
 I (3) ' P ::= T-TERMINARE; C – CONTINUARE; ' ECRIRE I' '
 I (3) 'JI : NUMAR JUDET DE INCEPUT; ' ECRIRE I ''
 I (3) 'JF : NUMAR JUDET SFIRSIT' I (+1) 'SAU SPATIU;'ECRIRE I''
 I (3) 'LLLLL : NUMAR LOCALITATE IN' I (+1) 'JUDETUL JI SAU SPATIU'
   ECRIRE I''
 M Z1 = U M Z1 = EXT
 M Z2 = U M Z2 = Z1 1 1
 SI Z2 ¬= 'C' ET Z2 ¬= 'T'
   ALORS
     I (1) '** ERR.FF.PARAMETRUL ⟨P⟩ ? ' ECRIRE
     REFAIRE
   SINON
     SI Z2 = 'T'
       ALORS
```

```
            I 'LA' REVEDERE I'
            SORTIE
          FIN
FIN
M Y1 = Z1 2 2 /* <JI> */
M Y2 = Z1 4 2 /* <JF> */
M Y3 = Z1 6 5 /* <LLLLL> */
SI Y1 = U OU Y1 <O OU Y2 < O OU Y3 < O
  ALORS
    I (1) '**ERR.FF.PARAMETRUL ... ? 'ECRIRE REFAIRE
FIN
SI EXISTE Y2 ET Y2 < Y1
    ALORS
    I (1) '**ERR.FF.PARAMETRUL <JF> ? 'ECRIRE REFAIRE
FIN
SI Y1 > 0 ALORS M Y1 = Y1 - 1 FIN
SI PAS Y3 ALORS M Y3 = 0 FIN
SI PAS Y2 ALORS M Y2 = 0 FIN
DEPUIS Y1
  POUR TOUT JUDETE X1
    M Y4 = NUMDE X1
    SI Y2 > 0
          ALORS
          SI Y4 > Y2 ALORS SORTIE FIN
    FIN
          M Y5 = 0
          I (2 - 2) Y4 I (+1) DEN1 I (+0) DEN2 ECRIRE I''
          POUR TOUT JUD-LOC X2 M Y5 = Y5 + 1
          SI Y5 < Y3 ALORS SUIVANT SINON M Y3 = 0 FIN
            M Y6 = Y5 / 90
            M Y7 = Y6 * 90 - Y6
            SI Y7 = 0
              ALORS I' '
                I (1) '* CKP:<LLLLL>= '
                I (+6 -5) Y5 ECRIRE I' ' /* ? */
            FIN
            I (6 - 5) COD I (+1) DEN1 I (+0) DEN2 ECRIRE
        FIN
    FIN
  REFAIRE
FIN ?
```

Comanda D

Rol : desemnarea formală a unei variabile X_i ca adjectiv

Sintaxa :

1) D X_i

În acest caz variabila X_i primește valoarea nedefinit. Această comandă este similară cu M $X_i = U$.

2) D X_i = UN ⟨*nume entitate*⟩ [⟨*calificare*⟩]

Prin această cerere variabila X_i va desemna formal ⟨*nume entitate*⟩. În acest caz X_i devine adjectiv de desemnare sau calificator. Pentru a putea utiliza această variabilă X_i ca marcator al unei realizări de entitate este necesar să se atribuie o adresă virtuală a acelei realizări.

Această cerere introduce un criteriu de validare contextuală a utilizării variabilei X_1.

Exemplu :

```
D X1 D X2
D X1 = UN PERSOANA
POUR TOUT LOCALITATI X2
   POUR TOUT LOCN-PERS X1    /* X1 utilizat în context  */
         I NUME              /* PERSOANA                */
   FIN
FIN ?
```

Comanda BLOQUER ... LIBERER

Rol : blocarea (BLOQUER) și deblocarea (LIBERER) consolelor (utilizatorilor !) active simultan pe aceeași bază de date, pentru a se asigura integritatea și acuratețea datelor introduse la creare/actualizare.

Sintaxa :

BLOQUER
  ⟨*suita de cereri*⟩ /* blocarea se realizează pe întreaga */
LIBERER /* perioadă de execuție a acesteia */

Specificații de utilizare :

1 – cererile BLOQUER...LIBERER nu pot fi imbricate ;

2 – pentru limitarea timpilor de blocare a consolelor active simultan este indicat ca cererile cuprinse între BLOQUER și LIBERER să nu fie de forma M ...= EXT (reclamă un dialog cu utilizatorul care poate deveni indisponibil, la un moment dat, sau este lent) sau LIRE DANS... (citirea unui terminal într-un buffer) ;

3 – indiferent dacă utilizatorul folosește o tehnică proprie de blocare sau nu este imperios necesar să se blocheze cererile care acționează asupra „resurselor alocabile" ale bazei de date. Aceste cereri (și „resursele alocabile" solicitate) sînt :

– comenzile de generare și/sau suprimare pe șiruri de biți, realizări de entitate (pot conține eventuale „resurse alocabile") și caracteristici de tip ⟨*inel*⟩ ;

– comenzile de actualizare a caracteristicilor de tip ⟨*referire*⟩.

Comanda LIRE/ECRIRE articole din fișiele secvențiale pe suport magnetic

Rol : citirea/scrierea unui articol într-un/dintr-un buffer de memorie de pe/pe suport magnetic (bandă sau disc).

Sintaxa :

1. *Citire* :

LIRE ⟨*identificator periferic*⟩ ⟨*număr periferic*⟩
  DANS ⟨*identificator buffer*⟩ ⟨*număr buffer*⟩

2) *scriere* :

ECRIRE ⟨*identificator buffer*⟩ ⟨*număr buffer*⟩
  DANS ⟨*identificator periferic*⟩ ⟨*număr periferic*⟩

⟨*identificator periferic*⟩ : nume generic atribuit perifericului de către programator ;
⟨*identificator buffer*⟩ : nume generic atribuit bufferului de către programator ;

Aceste nume generice sînt utilizate numai în scopul măririi lizibilității și semanticii comenzilor (nu sînt controlate de sistem).
<număr periferic> : număr întreg care reprezintă numărul logic al perifericului (pentru SOCRATE). Acest număr trebuie să fie identic cu acela declarat în comanda de conectare a unui *<număr buffer>* : număr întreg din intervalul [0,5] care reprezintă numărul logic al bufferului utilizat.
Specificații de utilizare :
— bufferul utilizat, desemnat prin *<număr buffer>*, pentru efectuarea operațiilor de citire și/sau scriere trebuie să fie rezervat și asociat unui format logic ;
— utilizarea fișierelor pe suport magnetic impune definirea acestora printr-o cerere de conexiune (ATTACH).

Pentru versiunile V 1.5 și V.1.6.R conexiunea se realizează astfel :
— la utilizarea fișierelor pe bandă magnetică :
% [*<eticheta>*] ATTACH MT *<număr periferic>*, unde *<număr periferic>* aparține intervalului[00,99], și trebuie să fie utilizat, ca atare, în toate cererile de intrare/ieșire pe acest periferic ;
— utilizarea fișierelor pe disc magnetic este posibilă (în versiunea SOCBTCH perfect compatibilă SOCRATE V.1.5 realizată de specialiștii din cadrul C.C.E. al C.S.P.) prin cererea de conexiune exprimată cu ajutorul comenzii :
% [*<eticheta>*] ATTACH ADF *<număr periferic>*, unde
<număr periferic> ∈ [0,1] și trebuie utilizat ca atare în toate cererile de intrare/ieșire pe acest periferic ;
aceste comenzi sînt disponibile numai în „batch processing".

Exemplu :

Încărcarea nomenclatorului LOCALITĂȚI descris în baza de date SOCRATE (capitolul 3).
Programul va fi folosit pentru operațiile de creere/modificare/ștergere.

```
[comenzi de lansare a modului SOCBTCH și conectare la baza de date dorită]
%  ACTI TB:49 /* comanda de activare adăugare la structura bazei de date    */
%  RUN FN:D  /* apel procesor : compilatorul LMD                             */
D /* se efectuează adăugare la structura existentă a unei noi structuri      */
FORMAL LOCALIT */ adăugarea se referă la un <formal logic>                   */
  DEBUT
    COD-JUD DILATE 2 /* codul județului de aparține                          */
    COD-LOC DILATE 6 /* codul localității                                    */
    LOC-SUP DILATE 6 /* codul localității de care aparține ierarhic          */
    DEN1-LOC MOT 30 /* denumirea în clar                                     */
    DEN2-LOC MOT 10 /* a localității tratate                                 */
 FIN FIN ?
 %  DACTI TB:49 /* dezactivare nivel adăugare la structura                   */
 %  ATTACH MT03  /* conectare banda magnetică cu numărul logic               */
                 /* SOCRATE 3                                                */
 %  FILE PRM:INP,RCS:54,BFS:558,RCF:FIX
/* banda va fi utilizată numai în operatorii de citire (PRM:INP) și conține  */
/* articole de lungime fixă (RCF:FIX) cu lungimea de 54 octeți (RCS:54)      */
/* blocate cîte 10 BFS : (RCS+1) * 10 + 8)                                   */
 %  ASSIGN DV:MT33 /* numărul logic recunoscut de sistemul de operare        */
 %  TABEL FN:'NOMENCL-LOCALIT' /* numele fișierului                          */
/* în cazul în care este detectată o eroare logică în datele citite          */
/* acestea vor fi salvate pe fișierul 'LOCALITATI-ERR' descris astfel :      */
```

```
% ATTACH MTO5
% FILE PRM:OUT,RCS:54,BFS:558,RCF:FIX
% ASSIGN DV:MT55
% LABEL FN:'LOCALITATI-ERR'
% RUN FN:R /* apel procesor:compilator LMD */
D X1 D X2 D X3 M Z1 = U /* anulare context variabile de adresă şi Z1 */
M X1 = FORMAL LOCALIT DANS ARTICOL 1 /* rezervare şi formatare buffer 1 */
FAIRE /* ciclu de prelucrare */
  S X1 /* iniţializare (ştergere) buffer */
  LIRE FIS-LOC 03 DANS ARTICOL 1 /* citeşte un articol în zona ARTICOL */
  M Y1 = COD-JUD DE X1
  SI Y1 = 99 /* condiţia logică de „sfîrşit fişier" */
    ALORS
      ECRIRE ARTICOL 1 DANS FIS-ERR 05 /* pune sfîrşit logic fişier erori */
      SORTIE /* ieşire din ciclul de prelucrare lansat (FAIRE) */
FIN
M X2 = UN JUDETE Y1
SI PAS X2
  ALORS
    ECRIRE ARTICOL 1 DANS FIS-ERR 05
    M Z1 = '* ATENTIE/EXISTA ARTICOLE ERR.?' /* marcare existenţa erorii */
    REFAIRE /* reluare cu o nouă citire */
FIN
M X3 = U
M Y2 = U
M Y2 = LOC-SUP DE X1
SI EXISTE Y2        /* dacă există indicaţia ca localitatea curentă */
  ALORS             /* este subordonată ierarhic mă asigur de existenţa */
    FAIRE           /* nivelului superior */
      POUR UN LOCALITATI X3 AVEC COD = Y2; DE X0
    FIN
    SI PAS X3       /* dacă localitatea de nivel ierarhic superior nu există */
      ALORS         /* atunci o generez completîndu-i informaţiile disponibile */
        G UN LOCALITATI X3
        M COD DE X3 = Y2
        M LOC-JUD DE X3 = X2
      FIN
FIN
M Y3 = COD-LOC DE X1
M X4 = U
  SI PAS Y3    /* dacă codul localităţii curente este vid sau nenumeric */
   ALORS       /* atunci articolul este eronat şi marchez existenţa erorii */
    ECRIRE ARTICOL 1 DANS FIS-ERR 05
    M Z1 = '*ATENTIE ! EXISTA ARTICOLE ERR. ?'
    REFAIRE /* reiau cu o nouă citire */
  FIN
  FAIRE /* localitatea există ? */
    POUR UN LOCALITATI X4 AVEC COD =YI; DE XO.
  FIN
  SI PAS X4 /* dacă nu există o generez şi îi pun codul */
    ALORS
     G UN LOCALITATI X4
  M COD-DE X4 = Y3
  FIN
  POUR X4 /* pentru localitatea curentă */
    M Z2 = U M Z2 = DEN1-LOC DE X1
    SI Z2 = 'S'  /* dacă DEN1-LOC conţine numai litera S */
      ALORS   . /* voi angrena procesul de ştergere a acestei realizări */
        M COD = U /* anulez codul din dicţionar */
        M LOC-JUD = U /* anulez legătura cu judeţul */
        M LOC-SUP = U /* anulez, dacă există, legătură cu nivelul superior */
        M DEN1 = U M DEN2 = U /* anulez denumirea */
        S X4   /* şterg bitul corespondent din şirul de biţi de prezenţă */
```

```
   REFAIRE /* trec la tratarea unui nou articol              */
  FIN
  M LOC-JUD = U /* dacă operaţia nu este de ştergere atunci o  */
  M LOC-JUD = X2 /* tratez ca operaţie de modificare, de aceea */
  M LOC-SUP = U /* anulez legătura cu judeţul, cu localitatea de */
  SI EXISTE X3 /* nivel superior şi completez aceste legături  */
    ALORS    /* cu valorile citite în înregistrarea curentă    */
    M LOC-SUP = X3
  FIN
  M DEN1 = DEN1-LOC DE X1
  M DEN2 = DEN2-LOC DE X1
  FIN
 REFAIRE /* treci la tratarea unui nou articol */
FIN      /* sfîrşit ciclu de prelucrare       */
SI EXISTE Z1 ALORS I (1) Z1 ECRIRE FIN
/* dacă a fost marcată eroare de date anunţ acest lucru */
? /* sfîrşit program : angrenează compilarea efectivă a programului
  /* şi în cazul corectitudinii sintactice şi semantice a cererilor componente
  /* (care formează unităţi lexicale) lansează execuţia efectivă
```

Observaţie :

Acest exemplu poate fi rulat pe Mini efectuind următoarele schimbări :
- apelul procesorului de definire conform specificaţiilor acestuia ;
- inlocuirea comenzilor ATTACH prin comenzile ATTACHE, incorporate în programul de cereri ;
- apel compilator LMD pentru acest program.

Citirea unui terminal într-un buffer

Rol : încărcarea unui buffer cu datele citite de pe un terminal conform unei structuri logice (format) definite de utilizator.

Sintaxa :

LIRE DANS ⟨*identificator buffer*⟩ ⟨*număr buffer*⟩

⟨*identificator buffer*⟩ : nume generic atribuit bufferului de către programator pentru mărirea lizibilităţii comenzilor sale ;

⟨*număr buffer*⟩ : număr întreg în intervalul [0,5] care reprezintă numărul logic al bufferului ;

Specificaţii de utilizare :

În prelucrarea conversaţională :
- fiecărui terminal i se alocă în mod implicit bufferul cu numărul 0 (zero) de 79 caractere şi orice citare a acestui număr într-o comandă de citire executată în programul lansat la terminal va provoca citirea unei linii de ecran ;
- bufferele cu numerele [1,5] se rezervă explicit de programator şi orice comandă de citire care citează unul din aceste buffere va fi adresată (şi executată de) către dispozitivul de intrare standard al partiţiei în care este lansat modulul SOCRATE conversaţional. In „batch proceesing" :
- toate bufferele se rezervă explicit de către utilizator iar comenzile de citire, cu această sintaxă, care citează aceste buffere se adresează (şi execută de) către dispozitivul de intrare standard al partiţiei.

Scrierea unei linii la terminal (în conversaţional) se realizează cu ajutorul comenzii ECRIRE pe buffer completat cu informaţii cu ajutorul comenzii I formatat.

Toate cererile de execuţie a unei comenzi I, forma de editare standard, se vor executa şi adresa implicit la terminal (în „batch processing“, care reprezintă o simulare a conversaţionalului, ieşirea standard a partiţiei este considerată terminal de ieşire iar intrarea standard terminal de intrare simulînd astfel, cu două dispozitive, un terminal conversaţional).

Modificări ale LMD-SOCRATE în V 1.6.R.

Aceste modificări au fost efectuate pentru a permite tratarea, pe de o parte, noilor tipuri de caracteristici ale LMD şi/sau a extensiilor caracteristicilor existente iar, pe de altă parte, pentru a introduce noi facilităţi de exploatare a tehnicii de calcul şi a uşura activitatea de programare.

Calificare prin DONT

A fost introdusă pentru a permite calificarea unei caracteristici aflate la nivel superior unei entităţi imbricate printr-o caracteristică aflată în entitatea imbricată.

Exemplu :

```
ENTITE    (n1) A1
  DEBUT
   CA1    MOT
  ENTITE  (n2) A2
   DEBUT
   CA2    MOT
  ENTITE  (n3) A3
   DEBUT
   CA3    MOT
   FIN
  FIN
FIN ?
```

Presupunem că în X1 se află adresa unei realizări a entităţii A3 ; putem califica o caracteristică din entitatea A2 prin X1 astfel :

I CA2 DE UN A2 DONT X1 ?

Pentru a califica o caracteristică dintr-o entitate la nivel superior nu este necesară specificarea entităţilor care includ structural entitatea de nivel inferior prin care se face calificarea cu DONT :

I CA1 DE UN A1 DONT X1

Forma generală a utilizării calificatorului DONT :

⟨caracteristică elementară⟩ [*⟨calificare⟩*] DONT *⟨citaţie de realizare de entitate imbricată la nivel inferior⟩*.

Exemplu :

I CA1 DONT UN A3 AYANT CA3 = 'CALIFIC'; DE UN A2 DE UN A1

Variabile de lucru W_i

W_i (i=1,14) reprezintă *⟨variabila de lucru⟩* pentru caracteristicile de tip DECIMAL (*zecimal împachetat cu virgulă*).

Sintaxa și modul de utilizare a variabilelor W_i sînt date de acțiunea al cărei obiect sînt și de tipul celorlalte obiecte implicate în această acțiune, astfel:
a) *atribuiri*:
1) M W_i = EXT [('*⟨constantă alfanumerică⟩*'/Z_i)]
Această comandă angrenează un dialog la terminal, cu utilizatorul, astfel:
– dacă în membrul drept este specificată ('*⟨constantă alfanumerică⟩*'/Z_i) atunci la terminal apare (dacă în sintaxă este utilizat Z_i atunci Z_i trebuie poziționat, anterior utilizării sale, la valoarea *⟨constantă* alfanumerică⟩):
⟨constantă alfanumerică⟩:
– dacă membrul drept este numai EXT la terminal apare:
W_i:
– indiferent de modul de inițiere al dialogului se așteaptă unul din răspunsurile:
/, NON, ⟨cr⟩: conținutul variabilei W_i rămîne neschimbat;
U: variabila Wi este pusă la nedefinit
⟨valoare⟩: dacă valoarea este corectă atunci se va atribui variabilei W_i, altfel se repetă dialogul.
⟨valoare⟩ : : = *⟨constantă numerică⟩*/*⟨constantă zecimală⟩*
⟨constantă zecimală⟩ : : = *⟨constantă numerică⟩*, *⟨număr⟩*/
[*⟨semn⟩*] *⟨număr⟩*$_1$, *⟨număr⟩*$_2$
⟨semn⟩ : : = Ѣ/ + / – /
Ѣ/ + : număr zecimal pozitiv
– : număr zecimal negativ
⟨număr⟩$_1$: partea întreagă a numărului zecimal (maxim 15 cifre);
⟨număr⟩$_2$: partea fracționară a numărului zecimal (maxim 7 cifre).
2) *M* W_i = *U* W_i primește valoarea nedefinit;
3) Conversii
– *⟨zecimal despachetat⟩-⟨zecimal împachetat⟩*:
M W_i = Z_i[[(]*⟨poziție⟩* *⟨lungime⟩* [)]]/
⟨constantă numerică⟩ /
⟨constantă zecimală⟩
Dacă Z_i este spațiu, nedefinit sau nu conține o valoare de tip *⟨constantă numerică⟩*/ *⟨constantă zecimală⟩* atunci W_i primește valoarea nedefinit altfel valoarea din membrul drept este convertită în zecimal împachetat (cu virgulă).
M Z_i [(] *⟨poziție⟩* [)] = W_i
– *⟨binar⟩* ↔ *⟨zecimal împachetat⟩*
M W_i = Y_i

Valoarea binară conținută de Y_i este transformată în zecimal împachetat (partea fracționară este nulă).

M Y_i = W_i

Dacă valoarea conținută de W_i este în afara intervalului $[-(2^{31}-1), +(2^{31}-1)]$ atunci Y_i este pus la valoarea nedefinit.

4) Expresii
M W_i [(K)] = *⟨expresie aritmetică⟩*
⟨expresie aritmetică⟩ : : = *⟨expresie aritmetică⟩*$_w$ /*⟨expresie aritmetică⟩*$_y$
⟨expresie aritmetică⟩$_w$: este o *⟨expresie aritmetică⟩* (§ 4.) în care

⟨*operand*⟩ : : = ⟨*număr*⟩/⟨*număr*⟩, ⟨*număr*⟩/W_i ;
⟨*expresie aritmetică*⟩$_y$: este o ⟨*expresie aritmetică*⟩ în care
⟨*operand*⟩ : : = ⟨*număr*⟩/⟨*număr*⟩, ⟨*număr*⟩/Y_i.
– Dacă în sintaxă este utilizat (*k*) atunci *k* specifică numărul zecimal la care să se facă rotunjirea ($k \in [0,6]$). Rotunjirea se face astfel :
– se compară cifra ($k + 1$) a părții zecimale cu 5 ;
– dacă cifra ($k + 1$) este mai mare atunci se adună 1 la cifra din poziția *k* (eventualul report de cifră semnificativă este propagat corespunzător) și se pun pe o (zero) pozițiile ($k + 1$) la 7 ;
– dacă cifra ($k + 1$) este mai mică sau egală atunci nu se efectuează rotunjirea.

Deși modul de definire a expresiei aritmetice arată acest lucru specificăm că utilizarea variabilelor W_i și Y_i în aceeași expresie este interzisă.

5) alte atribuiri
– *M* $W_i = W_k$ i, $k = 1,14$
– *M* W_i = {⟨*valoare numerică*⟩/⟨*zecimal*⟩/⟨*binar*⟩/
⟨*zecimal împachetat*⟩/⟨*zecimal despachetat*⟩}⟨*calificare*⟩
– *M* {⟨*valoare numerică*⟩/⟨*zecimal*⟩/⟨*binar*⟩/⟨*zecimal împachetat*⟩/
⟨*zecimal despachetat*⟩} [⟨*calificare*⟩] = W_i

Tabela **4.5. Comparații admise la comanda „SI"**

stinga \ dreapta	MOT	listă de valori	Z_i	MOT formal	caracteristică numerică	Y_i	BINAIRE formal	DECIMAL	W_i	PACKE DILATE formal	constantă alfanumerică	constantă întreagă	constantă zecimală	U'	EXISTE PAS
MOT	*	*	*	*							*			*	*
listă de valori	*	*	*	*							*			*	*
Z_i	*	*	*	*							*			*	*
MOT formal	*	*	*	*							*	*		*	*
caracteristică numerică					*	*	*					*		*	*
Y_i					*	*	*					*		*	*
BINAIRE formal					*	*	*					*		*	*
caracteristică DECIMAL								*	*	*		*	*	*	*
W_i								*	*	*			*	*	*
DILATE PACKE formal					*	*		*	*				*	*	

b) *Condiții*

Atunci cînd W_i se află în ⟨*membrul stîng*⟩ al unei ⟨*condiții simple*⟩ :
⟨*membru drept*⟩ : : = W_i/⟨*valoare numerică*⟩/⟨*zecimal*⟩/⟨*constantă alfanumerică*⟩/
⟨*constantă numerică*⟩/⟨*constantă zecimală*⟩/U

⟨*constantă alfanumerică*⟩ trebuie să fie de forma :

[⟨*semn*⟩] ⟨*număr*⟩[,⟨*număr*⟩]

Este admisă utilizarea lui W_i în ⟨*condiții de existență*⟩ :

{EXISTE/PAS} W_i

Pentru V 1.6.R a fost extinsă gama comparațiilor admise între tipurile de obiecte manipulate de LMD în conformitate cu extensiile LDD și LMD. În tabelul 4.5 sînt prezentate comparațiile admise la exprimarea cererilor condiționale iar în tabelul 4.6 comparațiile admise la comanda M.

Tabelul 4.6. **Comparații admise la comanda „M"**

dreapta / stînga	MOT	listă de valori	Z_i	MOT formal	caracteristică numerică	Y_i	BINAIRE formal	DECIMAL	W_i	PACKE DILATE formal	constantă alfanumerică	constantă întreagă	constantă zecimală	U	EXISTE PAS
MOT	*	*	*	*							*			*	*
listă de valori	*	*	*	*							*			*	*
Z_i	*	*	*	*		*			*		*			*	*
MOT formal	*	*	*	*							*			*	
caracteristică numerică					*	*	*	*	*	*		*		*	*
Y_i			*		*	*	*	*	*	*		*		*	*
BINAIRE formal					*	*		*	*			*		*	
caracteristică DECIMAL					*	*	*	*	*	*		*	*	*	*
W_i			*		*	*	*	*	*	*		*	*	*	*
DILATE PACKE formal								*	*	*		*	*	*	*

Accesul direct prin criteriu

Pentru caracteristicile de tip ⟨*cuvînt*⟩ declarate chei de acces discriminante este admisă :

1) selectarea realizărilor entității, clasate în dicționar, care conțin valori (atribuite cheii) care încep cu un (sub)șir de caractere stabilit de utilizator.

Sintaxa de utilizare este :

UN/TOUT} {⟨*nume entitate*⟩/⟨*nume invers*⟩} [X_i] AVEC
⟨*criteriu*⟩ [⟨*filtru*⟩] [⟨*calificare*⟩]

unde :

⟨*criteriu*⟩ : : = ⟨*identificator*⟩ [⟨*calificare*⟩] = = {Z_i]' ⟨*(sub)şir caracter*⟩' –}

⟨*identificator*⟩ : identificatorul unei caracteristici de tip ⟨*cuvînt*⟩ declarată cheie de acces discriminantă ;

= = : operatorul care indică faptul că selectarea realizărilor se va efectua comparînd doar primele ⟨*n*⟩ caractere din valoarea atribuită caracteristicii, ⟨*n*⟩ reprezentînd lungimea (sub/şirului (sau Z_i) utilizat în expresie ;

⟨*(sub)şir caractere*⟩ : valoarea de selecţie a realizărilor. Dacă în sintaxă este utilizat Z_i atunci Z_i trebuie poziţionat anterior utilizării sale la valoarea ⟨*(sub)şir caractere*⟩.

Exemplu :

Imprimarea tuturor bărbaţilor (nume şi prenume) din baza de date născuţi în anul 1952 :

```
POUR TOUT PERSOANA AVEC COD = ='152' ; DE X0
    I (1) NUME I (+1) PRENUME ECRIRE
FIN ?
```

2) utilizarea condiţiilor compuse pentru definirea criteriului de selecţie :

⟨*criteriu*⟩ : : = ⟨*condiţie*$_1$⟩ ET ⟨*condiţie*$_2$⟩/⟨*criteriu*⟩ ET ⟨*condiţie*$_e$⟩

⟨*condiţie*⟩$_e$: : = ⟨*caracteristică rapidă*⟩ = ⟨*valoarea comparare*⟩

⟨*condiţie*$_1$⟩ : : = ⟨*caracteristică rapidă*⟩{⟩/⟨/ = }⟨*valoare comparare*⟩

Din această sintaxă rezultă :

- este admisă o singură condiţie simplă cu operatorul de comparare '⟩' (mai mare) sau '⟨' (mai mic) şi aceasta trebuie să fie definită prima în cadrul condiţiei compuse ;
- între ⟨*valoare comparare*⟩ şi ⟨*caracteristică rapidă*⟩ trebuie să existe o identitate de tip ;
- ⟨*caracteristică rapidă*⟩ : : = ⟨*identificator*⟩ [⟨*calificare*⟩], ⟨*valoare comparare*⟩ : poate fi o valoare imediată, o variabilă de lucru sau identificatorul unei caracteristici din baza de date.

Exemplu :

Imprimarea tuturor femeilor născute după 1.12.1965 al căror nume începe cu 'PO' :

```
I TOUT PERSOANA AVEC COD > '2651201010010' ET
                     NUME = = 'PO' ;  DE  X0 ?
```

Definirea constantelor alfanumerice

Deşi SGBD-SOCRATE a fost conceput pentru lucrul în conversaţional nu dispune de comenzi care să-i permită utilizatorului gestionarea ecranului unui dispozitiv video conform dorinţelor sale (versiunea operaţională la noi a SGBD-SOCRATE a fost realizată în anii 1973–1975 cînd însăşi teoria teleprelucrării era în stadiul de definitivare).

Pentru a permite gestiunea unei varietăți mai mari de terminale s-a introdus facilitatea de a defini ⟨*constante alfanumerice*⟩ care să admită succesiuni de caractere exprimate în hexazecimal. Aceste caractere „speciale" reprezintă instrucțiuni și/sau caractere de control care permit ștergerea ecranului, inversiune video, poziționarea cursorului, protejarea cîmpurilor etc. din setul de caractere și/sau instrucțiuni ale terminalului sau protocolului de comunicație utilizat.

Sintaxa de definire a constantelor alfanumerice este :

⟨*constantă alfanumerică*⟩ : : = ⟨*caracter*⟩/⟨*constantă hexa*⟩/
⟨*caracter*⟩ ⟨*constantă alfanumerică*⟩
⟨*constantă hexa*⟩ : : = ⟨⟨*șir hexa*⟩⟩
⟨*șir hexa*⟩ : : = ⟨*caracter hexa*⟩/⟨*caracter hexa*⟩ ⟨*șir hexa*⟩
⟨*caracter hexa*⟩ : : = ⟨*cifră hexa*⟩⟨*cifră hexa*⟩
⟨*cifră hexa*⟩ : : = 0/1/2/3/4/5/6/7/8/9/A/B/C/D/E/F

Sintactic începutul unei ⟨constante hexa⟩ este detectat prin prezența caracterului „<" iar sfîrșitul prin prezența caracterului '>'. Acești delimitatori nu fac parte din ⟨*constanta alfanumerică*⟩. Numărul de cifre hexa definite în cadrul unei ⟨*constante hexa*⟩ trebuie să fie par. Orice eroare detectată în cazul definirii unei constante hexa este de tip avertisment (WARNING).

Acest tip de constante pot fi atribuite, în batch sau conversațional, variabilelor de lucru Zi, caracteristicilor de tip ⟨*cuvînt*⟩ sau ⟨*text*⟩ și pot fi manipulate cu ajutorul comenzilor „editorului de texte" SOCRATE.

Exemple :

– constantei '⟨270F⟩ABCD' îi corespunde șirul hexazecimal (EBCDIC) '270FC1C2C3C4' ;
– constantei '10⟨270F⟩⊔⊔ȘTERGERE ECRAN' îi corespunde șirul hexazecimal 'F1F0270F4040E2E3C5D9C7C5D9C540C5C3D9C1D6' (EBCDIC).

Comanda M

1) Posibilitatea substituirii imprimării standard, la cereri de tip M. . . = EXT, cu un ⟨*șir de caractere*⟩ desemnat de utilizator :
M {⟨*variabilă de lucru*⟩/⟨*citație caracteristică elementară*⟩} =
(Z_i/'⟨*constantă alfanumerică*⟩')

Imprimarea {⟨*variabilă de lucru*⟩/⟨*identificator*⟩} : este substituită cu ⟨*constantă alfanumerică*⟩ :. Dacă este utilizat, în sintaxa cererii, Z_i atunci Z_i trebuie poziționat, anterior utilizării sale, la valoarea '⟨*constantă alfanumerică*⟩' (poziționarea se poate realiza și atribuindu-i valoarea conținută de un element oarecare al bazei de date respectînd evident corespondența de tip).

Exemplu :

– se dorește introducerea datei, sub forma ZZ LL AA, în variabila Y25 :
M Y25 = EXT ('DATA ZILEI SUB FORMA ZZ LL AA')

Pe ecran, sau imprimantă (batch) va apare, la execuția comenzii :

DATA ZILEI SUB FORMA ZZ LL AA : iar răspunsul utilizatorului, după validare, va fi atribuit variabilei Y25 ;

2) Manipularea caracteristicilor de tip ⟨*text*⟩ :
a) atribuirea ⟨*constantelor alfanumerice*⟩ :

M ⟨*identificator text*⟩ [⟨*calificare*⟩] = EXT dau naștere unui dialog cu operatorul de forma:

CREATION UN ⟨*identificator text*⟩:

Pe răspunsul OUI apare:

TEXTE: și se poate atribui, ca valoare, o ⟨*constantă alfanumerică*⟩ care conține un ⟨*șir hexa*⟩.

b) compararea cu caracteristici de tip ⟨*cuvînt formal*⟩ — M ⟨*identificator text*⟩ [({Y_i/⟨n⟩})][⟨*calificare*⟩] = ⟨*identificator cuvînt formal*⟩⟨*calificare*⟩

{Y_i/⟨n⟩}: desemnează numărul liniei din caracteristica de tip ⟨*text*⟩ la care se face referire. Dacă acest parametru nu este prezent atunci instrucțiunea M va acționa asupra primei linii vide care urmează ultimei linii încărcate cu modificarea corespunzătoare a controlului de linii;

– M ⟨*identificator cuvînt formal*⟩ [⟨*calificare*⟩] = ⟨*identificator text*⟩ [({Y_i/⟨n⟩})] ⟨*calificare*⟩

Dacă Y_i/⟨*n*⟩ nu desemnează o linie din intervalul [1, număr linii încărcate] se editează un mesaj de eroare:

c) numărarea liniilor nevide (încărcate) ale unei caracteristici de tip ⟨*text*⟩:

M Y_i = D ⟨*identificator text*⟩ ⟨*calificare*⟩

3) atribuirea unei ⟨*constante alfanumerice*⟩ care conține un ⟨*șir hexa*⟩ la o variabilă de lucru sau caracteristică elementară de tip ⟨*cuvînt*⟩

M Z_i /⟨*identificator cuvînt*⟩ [⟨*calificare*⟩] = ' ⟨*constantă alfanumerică*⟩'

Comanda I

În cazul utilizării editării cu format bufferul de ieșire a fost mărit la 132 caractere. Dacă dispozitivul de ieșire (imprimantă sau terminal, funcție de modul de prelucrare ales) admite un număr mai mic de caractere pe linie sistemul asigură „spargerea bufferului" și imprimarea pe linia (liniile) următoare.

1) definirea „șabloanelor de editare" pentru variabilele de lucru numerice (Y_i și W_i) și caracteristicile de tip ⟨*valoare numerică*⟩ sau ⟨*zecimal*⟩.

Sintaxa:

I [⟨A⟩]' ⟨*șablon de editare*⟩') {Y_i/W_i/⟨*citație caracteristică*⟩}

⟨*citație caracteristică*⟩: citația unei caracteristici de tip ⟨*valoare numerică*⟩ sau ⟨*zecimal*⟩.

⟨*sablon editare*⟩: șir de caractere de editare (similar COBOL) care semnifică:

9 : poziția unei cifre care va fi înlocuită cu cifra curentă din număr;

Z : poziția unei cifre care va fi înlocuită cu spațiu dacă este 0 (zero);

* : poziția unei cifre nesemnificative care va fi înlocuită cu '*' dacă este 0 (zero);

– : dacă numărul este negativ se pune minus în fața primei cifre semnificative, în caz contrar se lasă spațiu;

\+ : dacă numărul este pozitiv se pune semnul plus (+) iar dacă este negativ se pune semnul minus (–).

Aceste caractere de editare pot fi utilizate cu repetiție

. : poziția sa în sablon specifică poziția virgulei (marca zecimală).

Orice alt caracter, utilizat în expresia șablonului de editare, va reprezenta un caracter de inserție (va apare în textul imprimat între pozițiile definite de caracterele destinate reprezentării cifrelor).

⟨A⟩: reprezintă poziția din buffer de unde se efectuează imprimarea.

Dacă ⟨A⟩ nu este utilizat în sintaxă iar comanda nu face parte dintr-o secvență de comenzi I cu format (pentru a aparține unei astfel de secvențe este necesară pre-

zenţa unei comenzi I în care apare ⟨A⟩, definite înaintea acesteia, fără un ordin ECRIRE care să le separe) atunci comanda I devine de forma „editare standard" în care numele caracteristicii sau variabilei de lucru, care constituie obiectul acţiunii, nu mai este imprimat (se imprimă numai eventualele caractere de inserţie din sablon).

Exemple :

1) — M Y1 = 1101952
I ('DATA NAŞTERII 99/99/9999') Y1 ?
DATA NAŞTERII 01/10/1952 /* rezultatul imprimării */
2) imprimarea conţinutului unei (unor) linii al unei caracteristici de tip ⟨*text*⟩ :
I(⟨A⟩ [⟨B⟩]) ⟨*nume text*⟩ {Y_i/⟨n⟩} [⟨*calificare*⟩]

Dacă linia {⟨n⟩/Y_i} este încărcată (nevidă) atunci se va imprima din poziţia (⟨A⟩) (pe lungimea şi cu alinierea (⟨B⟩)) definită de utilizator conţinutul său. Dacă linia desemnată este vidă atunci se va semnala acest lucru printr-un mesaj de eroare.
3) specificarea unui caracter de umplere a bufferului (sau a unei porţiuni bine determinate a acestuia) de imprimare :
I (⟨A⟩[⟨*lg*⟩]) 'X'
⟨A⟩ : poziţia relativă în bufferul de ieşire ;
⟨*lg*⟩ ∈ [1,132] — lungimea zonei desemnate pentru a fi umplută cu caracterul 'X'
Dacă ⟨lg⟩ nu este prezent atunci i se atribuie valoarea 30. Bufferul de ieşire este umplut din poziţia desemnată de ⟨A⟩ pe lungimea ⟨*lg*⟩ cu caracterul 'X'.
4) specificarea octetului de salt la comanda ECRIRE :
ECRIRE [⟨n⟩]
Valorile atribuite lui ⟨*n*⟩ şi semnificaţia acestora sînt :
0 : imprimarea se realizează după saltul la pagina nouă ;
1—8 : imprimarea se realizează după un salt de 1—8 interlinii ;
9 : se realizează suprascriere.

În cazul utilizării acestei sintaxe în conversaţional rezultatul acţiunii comenzii este echivalent unui ECRIRE simplu.

Testarea condiţiilor de execuţie a instrucţiunilor LMD

La execuţia unui program SOCRATE se pot detecta erori în cazul :

— existenţei / inexistenţei obiectului acţiunii comenzii (realizări de entitate sau chei din dicţionar) la specificarea lui explicită ;
— actualizare sau controlul semantic al datelor (valori nenumerice, valori în afara limitelor, valori diferite de cele declarate în lista etc.) ;
— la apelul subprogramelor (program IMT expirat, model de expansiune incoerent (IMT şi PRO), lipsă program din bibliotecă etc.).

În funcţie de modul de utilizare, la detectarea unei erori, SOCRATE acţionează astfel :

— „*batch processing*" : se editează mesajul de eroare iar prelucrarea se desfăşoară în secvenţă (se poate evita imprimarea standard a mesajelor de eroare prin poziţionarea nivelului 78 cu comanda % ACTI TB : 78) ;

— *interfaţa cu programe evoluate* : se suspendă execuţia subprogramului SOCRATE în care s-a detectat eroarea cu returnarea numelui sau şi a codului de eroare la apelant ;

— *conversațional* : se editează mesajul de eroare iar execuția se derulează în secvență.

Pentru detectarea și eventuala construire a unor secvențe programator de depanare a erorii s-a introdus cuvîntul cheie **ERREUR** (care desemnează un registru special) a cărui valoare (număr întreg) și cauză a poziționării la această valoare poate fi :

— nedefinit (*U*) : nu s-a detectat nici o eroare ;

— 1 : FIN DE FICHIER — detectarea sfîrșitului fizic al unui fișier (sens SGF) înaintea celui logic (sens SOCRATE) ;

— 3 : REALISATION DEJA PRESENTE — tentativă de creere (generare) a unei realizări de entitate (desemnată prin număr de realizare) care există în baza de date ;

— 6 : ENTITATE SATURE — tentativă de creere (generare) a unei realizări de entitate pe o entitate saturată (toți biții din șirul de biți de prezență au valoarea 1) ;

— 7 : RANG ABSOLU'EN DEHORS ENTITE — tentativă de desemnare explicită a unei realizări în afara intervalului [1, Nmre] ;

— 9 : REALISATION ENTITE ABSENTE — tentativă de desemnare explicită a unei realizări de entitate care nu există (nu a fost creată, generată sau a fost ștearsă înaintea citării sale) ;

— 14 : CLE DEJA PRESENTE — tentativă de atribuire la o caracteristică declarată cheie de acces discriminantă (AVEC CLE UNIQUE) a (unei valori atribuite anterior unei realizări a acestei caracteristici realizare în sensul asocierii la o realizare de entitate) ;

— 15 : CLE NON TROUVE — tentativă de citare a unei realizări a unei caracteristici declarate cheie de acces discriminante după o valoare care nu este clasată în dicționarul asociat ;

— 82 : VALEUR NUMERIQUE ERRONEE — tentativă de atribuire (la o caracteristică de tip ⟨*valoare numerică*⟩ ⟨*zecimal*⟩ sau variabilă de lucru Y_i/W_i) a unei valori nenumerice sau valoarea numerică este în afara intervalului de variație declarat ;

— 84 : VALEUR DE LISTE ERRONEE — tentativă de atribuire la o caracteristică de tip ⟨lista de valori⟩ a unei valori nedeclarate în listă ;

— 128 : LANCEMENT PROGRAMME IMPOSIBILE — lansarea programului imposibilă (programe IMT sau precompilate cu erori în modelul de expansiune — distrus de incidente hard sau soft) ;

— 144 : PROGRAME PERIME — program perimat (pentru programele IMT s-a depășit data de expirare a valabilității).

Notă : desemnarea explicită a unei realizări de entitate se face prin utilizarea numărului de realizare ({număr/Y_i}).

Cuvîntul cheie ERREUR poate fi utilizat :

1) în exprimarea condițiilor simple :

⟨*condiție simplă*⟩ : : = *ERREUR* = {⟨*coderr*⟩/Y_i/*U*}

⟨*coderr*⟩/Y_i : codul erorii care interesează .

```
SI ERREUR = {⟨coderr⟩/Yi/U}
   ALORS
   ⟨secvența program⟩
   .
   .
   .
FIN
```

2) în exprimarea condițiilor de existență :
{EXISTE/PAS} ERREUR

Exemplu :

```
M Y1 = Z1
SI EXISTE ERREUR
  ALORS
    I (1) '*ERR.CONVERSIA IN BINAR A NR.'
    I (+1) Zi I (+1) 'IMPOSIBILA ?'
    ECRIRE
FIN
```

3) anularea registrului ERREUR :
M ERREUR = U

Întîlnirea cuvîntului cheie ERREUR, în cadrul unui program, provoacă inhibarea modului standard de editare a mesajelor de eroare și activează un mod propriu de editare a mesajelor de eroare care apar sub forma :

PR. *⟨prog⟩* [NR. *⟨nrrec⟩*] [NL. *⟨nrlin⟩*] [*⟨identificator⟩*] *⟨mesaj de eroare⟩*

⟨prog⟩ ; numele programului **MAIN** pentru un program principal care nu este sub formă de macroinstrucțiune sau program precompilat) în cadrul căruia s-a detectat eroarea ;

⟨nrrec⟩ : numărul de recursivitate al programului (numărul de imbricare conform instrucțiunii **ENCOURS**) ;

⟨nrlin⟩ : numărul liniei la care s-a detectat eroarea ;

⟨mesaj de eroare⟩ : mesajul de eroare în clar ;

⟨identificator⟩.

Citirea/scrierea unui terminal/buffer într-un buffer/pe un terminal

Pentru a extinde posibilitățile de utilizare a terminalelor de teletransmisie sintaxa comenzilor LIRE/ECRIRE a fost modificată astfel :

1) **LIRE** [{'*⟨antet mesaj⟩*'/Z_i}] **DANS** *⟨identificator buffer⟩* *⟨număr buffer⟩*

Citirea se efectuează pe întreaga lungime rezervată (prin placarea unui formal logic) pentru *⟨număr buffer⟩* permițînd astfel utilizarea terminalelor de tip display în „BLOCK MODE".

{'*⟨antet mesaj⟩*'/Z_i} : acest mesaj va fi afișat la terminal înaintea declanșării citirii efective.

2) **ECRIRE DE** *⟨identificator buffer⟩* *⟨număr buffer⟩*

Conținutul bufferului este scris fără transmiterea caracterelor de încheiere a transmisiei.

Definirea temporară a unei caracteristici de tip ⟨formal⟩

Compilatorul LMD (procesorul de cereri) a fost modificat pentru a accepta structuri de program de forma :

```
DEBUT
  ⟨lista formal⟩
FIN
  ⟨program de interogare⟩
  ?
```

⟨*lista formal*⟩ : caracteristici de tip ⟨*formal logic*⟩ ;

⟨*program de interogare*⟩ : secvența de instrucțiuni LMD care constituie corpul programului.

Sortare

Pentru utilizarea sortării programatorul are la dispoziție următoarele acțiuni :

1) rezervarea unui buffer pentru caracteristica ⟨*formal pentru sortare*⟩ utilizată, astfel :

M X_i = FORMAL SORT DANS ⟨*identificator buffer*⟩ ⟨*număr buffer*⟩

Specificații de utilizare :

– dacă bufferul ⟨*număr buffer*⟩ nu există se alocă ;

– dacă în bufferul ⟨*număr buffer*⟩ era rezervat de un formal normal (fără sortare) acest bufffer va fi refolosit numai dacă lungimea este mai mare sau egală cu lungimea bufferului de sortare. În caz contrar se alocă un nou buffer ⟨*număr buffer*⟩ ;

– dacă bufferul ⟨*număr buffer*⟩ era rezervat de un ⟨*formal pentru sortare*⟩ diferit de cel actual atunci această rezervare va fi anulată ;

– dacă bufferul ⟨*număr buffer*⟩ se rezervă pentru un formal normal atunci rezervarea pentru ⟨*formal pentru sortare*⟩ va fi anulată ;

– după rezervarea bufferului caracteristicile definite în ⟨*formal pentru sortare*⟩ pot fi modificate cu ajutorul comenzii M.

2) scrierea unui articol destinat sortării :

ECRIRE SORT DANS ⟨*identificator buffer*⟩ ⟨*număr buffer*⟩

Specificații de utilizare :

– dacă formalul pentru sortare este declarat cu cheie cu adunare (AVEC CLE ADDITION) atunci vor fi adunate toate cîmpurile de tip ⟨*binar*⟩ și ⟨*zecimal împachetat*⟩ din corpul articolului pentru toate articolele cu aceeași cheie ;

– variabila SORT va fi poziționată la valoarea :

4 : umplerea zonei de sortare ;

1 : după execuția fiecărei instrucțiuni de forma :

M . . . = ⟨*caracteristică cheie*⟩ DE X_i dacă valoarea atribuită acestei caracteristici din articolul curent este diferită de valoarea din articolul precedent.

⟨*caracteristică cheie*⟩ : identificatorul unei caracteristici declarate (membră) în corpul cheii.

3) citirea articolelor sortate :

LIRE SORT [AVEC POSITION] [{AVANT/ARRIERE}] –
DANS ⟨*identificator buffer*⟩ ⟨*număr buffer*⟩

Specificații de utilizare :

– în funcție de opțiunea utilizatorului citirea articolelor se va efectua în ordinea crescătoare (AVANT) sau descrescătoare (ARRIERE) a valorilor cheii ;

– declarația AVEC POSITION permite citirea articolelor, în ordinea de parcurgere cerută, începînd cu o anumită valoare a cheii. Această valoare de plecare a cheii trebuie atribuită, anterior citirii, corpului cheii (cu instrucțiunea M) ;

– după citirea articolului variabila SORT va fi poziționată astfel :

0 : cheia articolului citit este egală cu cheia articolului aflat anterior în buffer ;

1 : cheia articolului citit este diferită de cheia articolului aflat anterior în buffer ;

2 : s-a detectat sfîrșitul fișierului de sortare ;

— la momentul detectării sfîrşitului fişierului de sortare toate variabilele X_i folosite de această sortare vor fi puse la nedefinit (X_i care identifică bufferul pe care este placat formalul pentru sortare) ;

— la apelul unor subprograme care utilizează ⟨*formal pentru sortare*⟩ trebuiesc transmişi subprogramului toate variabilele X_i care identifică bufferul pentru sortare ;

— variabila **SORT** poate fi utilizată numai în condiţii.

Cereri de constituire a punctelor de control

a) Comanda PAUSE

SGBD-SOCRATE permite, la nivelul limbajului de comandă în prelucrarea conversaţională, constituirea unui punct de control la cererea expresă a operatorului de la terminal prin comanda **PAUSE** (3). Această cerere de constituire se realizează în mod „manual" şi este la dispoziţia operatorului (nu face parte din logica programului).

Pentru a permite automatizarea luării deciziei de construire a unui punct de control şi a lega această decizie de logica intrinsecă a programului de interogare s-a pus la dispoziţia programatorului de aplicaţie comanda PAUSE. Constituirea efectivă a punctului de control se realizează numai dacă este activă o comandă CHECKPOINT(CKPT).

Sintaxa :

PAUSE Y_i

Variabila Y_i este poziţionată astfel :

— $Y_i = 0$: nu există nici-o cerere de constituire a unui punct de reluare (Checkpoint) activă ;

— Y_i = ⟨*nr*⟩ : ⟨*nr*⟩ reprezintă numărul CKPT activ.

Pentru $Y_i \neq 0$ utilizatorul trebuie să memoreze contextul logic de execuţie al programului său într-o realizare de entitate (de exemplu numită LOG) al cărei număr este reprezentat de numărul utilizatorului (numărul realizării entităţii UTILISATEUR, din baza de baze, în care sînt memorate datele de identificare ale utilizatorului). Acest număr este furnizat de către sistem în variabila Y25 la realizarea conexiunii (LOGIN) cu baza de date şi nu este permisă modificarea sa în program.

Exemplu :

Presupunem că la introducerea fiecărei PERSOANA, în baza de date, s-a prevăzut constituirea unui punct de control. Pentru a memora contextul logic de execuţie s-a utilizat entitatea LOG care conţine cîmpul MATRICOL în care se va memora codul persoanei. În cadrul prelucrării curente codul persoanei este disponibil în variabila Z1. Secvenţa de execuţie a comenzii PAUSE este :

```
PAUSE Y1 SI Y1¬=0
          ALORS
             M MATRICOL DE UN LOG Y25 DE X0 = Z1
             M ULTIMUL-CKPT DE UN LOG Y25 DE X0 = Y1
             PAUSE
    FIN
```

Programul de încărcare trebuie :

— să înceapă cu secvenţa :

```
SI EXISTE UN LOG Y25 DE X0
   ALORS
```

```
    I (1) 'PRIMA PERSOANA DE INTRODUS:'
    I (+1) MATRICOL DE UN LOG Y25 DE X0
    I (+2) '(ULTIMUL CKPT DE PERSOANA: ' I (+1) Y1 I(+1) ')'
    ECRIRE
  FIN
```

– să se termine cu secvența :

S UN LOG Y25 DE X0

b) **Comanda CKPT**

Deoarece constituirea unui punct de control cu comanda PAUSE este efectivă numai dacă este activă o comandă CKPT s-a oferit posibilitatea activării necondiționate a cererii de constituire a unui punct de control prin comanda : CKPT.

Compilarea condițională a programelor (DEBUGGING)

Procesoarele (0(R) și M admit secvențe de program condiționale a căror tratare este realizată la specificarea parametrului DBG pe comanda % RUN FN : $\left\{ \begin{matrix} O \\ M \end{matrix} \right\}$, DBG (în „batch processing") s-au prin apelarea în conversațional a procesoarelor prin $ GCDB$\left\{ \begin{matrix} O \\ M \end{matrix} \right\}$în locul comenzii $GO.

Secvențele condiționale sînt specificate prin :

/< – început secvență ;

>/ – sfîrșit secvență.

Programele precompilate își păstrează caracteristica de DEBUGGING la apelare în funcție de modul cum au fost compilate (cu DBG sau fără). În cazul execuției unui program precompilat, cu opțiunea DBG, la momentul începerii secvenței condiționale se emite mesajul :

PR. ⟨*p*⟩. NR ⟨*n*⟩ NL ⟨1⟩ **** TRACE DBG unde :

– ⟨*p*⟩ : numele programului (programului principal i se atribuie numele MAIN) ;

– ⟨*n*⟩ : numărul de recursivitate (conform instrucțiunii ENCOURS – la macrogenerator) ;

– ⟨1⟩ : numărul liniei din programul principal la care începe secvența DBG. În cazul detectării unei erori la execuție (la utilizarea DBG) mesajul de eroare va conține și numărul liniei din cadrul programului.

Compilarea textelor programelor de cereri

Ca și în limbajul de definire (LDD), în limbajul de manipulare a datelor (LMD) scrierea liniilor sursă se face de manieră naturală iar problema ambiguității între un identificator și un cuvînt cheie este rezolvată rezervînd cuvintele cheie.

Modul de funcționare al compilatorului programelor de cereri este similar cu cel al LDD cu deosebirea că analiza depinzînd de context, în cazul citațiilor, se face nu numai asupra întregii fraze ci și asupra textului de definire la care se raportează. Pentru prezentare vom face apel la tabelele rezervate compilatorului anunțate în capitolul 3.

Principalele verificări făcute de compilatorul LMD, în raport cu structura de referinţă, sînt :

— verificarea definirii identificatorilor, care apar în textul citaţiei, în textul de definire ;

— verificarea tipului identificatorilor :

— • verificarea prezenţei calificatorului DE/POUR pentru ⟨*entitate*⟩ şi ⟨*bloc*⟩ ;

• verificarea existenţei caracteristicilor, indicate în citaţie, în cadrul caracteristicii de tip ⟨*referinţe*⟩/⟨*inel*⟩ cînd acest tip de caracteristică este utilizată drept calificator ;

— verificarea ierarhiei indetificatorilor (pentru a cita o caracteristică oarecare din baza de date trebuie să menţionăm în citaţie identificatorii tuturor blocurilor şi entităţilor intermediare) ;

— verificarea prezenţei cuantificatorilor {{UN/UNE}/{TOUT/TOUTE}} numai înaintea identificatorilor de ⟨*entitate*⟩ ;

— verificarea filtrelor şi condiţiilor :

• numai identificatorii de entitate pot fi urmaţi de AYANT/EXISTE/PAS/TELQUE ;

• în filtrele introduse se verifică dacă identificatorul plasat imediat în stînga operatorului din expresiile booleene reprezentate este o caracteristică care aparţine entităţii căreia i se aplică filtrul (identificatorul care precede lui AYANT/EXISTE/PAS/TELQUE/AVEC) ;

— verificarea conformităţii de ⟨*tip*⟩ a părţii din ⟨*stînga*⟩ şi a celei din ⟨*dreapta*⟩ a unui operator :

• o entitate nu poate fi comparată decît cu o altă entitate sau o caracteristică de tip referire ;
• operaţiile aritmetice nu sînt permise decît între caracteristici numerice, caracteristici care fac referire la caracteristici numerice sau valori numerice directe ;
• nu sînt permise comparaţiile directe între caracteristici de tip ⟨*bloc*⟩ şi ⟨*text*⟩ ;

— verificarea calificatorilor de tip variabilă X_i ;

• verificarea dacă numele după care apare adjectivul (X_i) este nume de entitate ;
• realizarea legăturii între adjectivul de desemnare şi adjectivul apel ;
• împiedicarea definirilor ambigue ;
• definirea domeniului de portabilitate a fiecărui adjectiv.

Pentru a realiza codul intern al citaţiilor compilatorul execută două treceri :

1) *analiza* (*sens direct*) care constă în :

— suprimarea cuvintelor cheie DE/POUR care apar înaintea entităţilor şi blocurilor ;

— codificarea cu un cod intern a cuvintelor cheie ;

— reprezentarea identificatorilor utilizaţi în citaţie printr-un punctator în DICONOM ;

— reprezentarea structurii logice a citaţiei indicînd :

• lungimea sa ;
• un punctator spre entitatea filtrată plasat la sfîrşitul filtrului în locul lui ';'

— codificarea adjectivelor, după verificare, în funcţie de utilizarea lor ;

— codificarea operatorilor (=, +, /, ...) printr-un cod intern.

2) *preinterpretarea (sens structural logic)* :

– căutarea caracteristicilor structurii interne asociate identificatorilor utilizaţi în citaţie :

- verificare tip, ierarhie ;
- înlocuirea punctatorilor spre DICONOM cu adresele virtuale ale caracteristicilor identificate în structura internă ;

– verificarea conformităţii sintactice între partea stîngă şi partea dreaptă a unui operator, în cadrul filtrelor şi modificarea codului operatorilor în funcţie de tipul părţii din dreapta ;

– definirea domeniului de portabilitate a adjectivelor de desemnare şi apel.

Codul intern obţinut este memorat într-o pilă de lucru (în partea de jos a lui BROU) şi este şters după ce sistemul răspunde acţiunii comandate. Pentru exemplificare, în tabelul 4.7 sînt prezentate cîteva din codurile interne utilizate pentru cuvintele cheie şi comenzile LMD în implementarea FELIX V 1.5/V 1.6.R.

Tabelul **4.7. Coduri interne utilizate de compilatorul LMD (FELIX C-***)**

COD HEXA	CUVÎNT CHEIE/ COMANDA	COD HEXA	CUVÎNT CHEIE/ COMANDA
4A	I	64	SORTIE
4B	S	65	ECRIRE
4C	G	66	REFAIRE
4D	C	67	EXEC
4E	M	6F	SI, PAS, EXISTE
4F	POUR, DEPUIS	75	definire X,
50	SE	76	definire context
60	FAIRE	77	început expansiune program
61	DEPUIS	78	LIBERER
62	EXT	79	BLOQUER
63	SUIVANT		

Codificarea internă a unei citaţii trebuie să permită interpretorului să o parcurgă în sens structural logic, adică mergînd de la „baza" sa spre „capul" său.

Concluzii

Acest capitol a fost destinat prezentării LMD SOCRATE şi conceptelor proprii acestuia.

Recomandăm cititorului să acorde o deosebită importanţă noţiunilor de citaţie, calificare şi filtru care dau, în general, complexitatea unui program. Mai mult, prin varietatea de exprimare a acestora, se poate aplica metoda rafinării în paşi succesivi a programului pentru a obţine o formă concisă de exprimare a acestuia.

Au fost prezentate, de asemenea, modalităţile de realizare a accesului la elementele conţinute în baza de date.

Diversitatea tipurilor de acces permise combinată cu posibilitatea părăsirii şi/sau combinării acestora prin simpla formă de exprimare a citaţiei elementelor permit acea „navigare" prin date anunţată la începutul capitolului.

O parte importantă a capitolului a fost destinată prezentării în detaliu a comenzilor LMD.

Pentru exemplele prezentate în acest capitol atragem atenţia că sînt valabile pe orice implementare a SGBD-SOCRATE, indiferent dacă conţin sau nu diverse „artificii" de programare. În cazul în care exemplele sînt prezentate împreună cu comenzi „batch processing" V 1.5 aceste comenzi vor fi înlocuite cu echivalentul lor, prezentat drept comentariu asociat liniei de comandă, în implementarea sub care se testează programele.

Au fost prezentate şi modificările LMD aduse de versiunea V 1.6.R. Ultimul paragraf oferă o imagine succintă a modului de compilare a textelor programelor de cereri. Scopul său este acela de a atrage atenţia utilizatorului asupra complexităţii verificărilor realizate de compilator şi pentru a-l ajuta să înţeleagă mai uşor de ce o frază „aparent corectă" este respinsă de compilator.

5. ALTE COMPONENTE ALE SGBD-SOCRATE

Macrogeneratorul

Scopul macrogeneratorului este acela de a furniza o prelungire a limbajului de cereri, care modifică, pentru o aplicaţie dată, limbajul indus de structură şi înţeles efectiv de compilator.

Macrogeneratorul transformă fiecare frază scrisă în limbajul definit într-o frază în limbajul real.

Acest lucru nu implică nici creşterea nici diminuarea posibilităţilor limbajului de cereri, ci o modificare a modului său de prezentare externă.

Modul general de funcţionare al macrogeneratorului este acela de a înlocui şiruri de caractere în conformitate cu reguli (care pot fi introduse în orice moment) definite de utilizator şi de a conserva eventual aceste reguli.

Regulile introduse de utilizator îi permit acestuia să definească o corespondenţă între un şir de caractere numit ⟨*nume macro*⟩ şi o suită de cereri, o cerere sau o parte bine determinată dintr-o cerere, care formează expansiunea macroinstrucţiunii.

Macrogeneratorul introduce două facilităţi importante :

— utilizatorul lucrează cu noul limbaj ca şi cum acesta ar fi limbajul sistemului, acţiunea macrogeneratorului fiind disimulată utilizatorului ;

— evită accesul neautorizat şi eventualele erori de manipulare a datelor din partea utilizatorilor nespecialişti.

Macrogeneratorul permite definirea de macroinstrucţiuni (macro), programe precompilate, ambele tipuri scrise în limbajul de manipulare a datelor, şi programe executabile (rezultate din programe scrise în limbaj de asamblare).

În afara cuvintelor cheie ale limbajului de manipulare a datelor, macrogeneratorul are un set de cuvinte cheie proprii :

⟨cuvînt cheie macrogenerator⟩ : : = : DEF {MAC/PRO/IMT}/
: SUP {MAC/PRO/IMT/EXP}/
: LIS {MAC/PRO/IMT/ALL}
: EDI {MAC/PRO}/ : EXP/ : PW/ : FDEF

unde componentele implicate la formarea cuvintelor cheie au semnificaţia :

— DEF :	defineşte modelul de apel ;	— MAC :	macroinstrucţiune ;
— SUP :	suprimă ;	— PRO :	program precompilat ;
— LIS :	listează ;	— IMT :	program în format IMT ;
— EDI :	editează ;	— ALL :	toate obiectele gestionate de macrogenerator ;
— EXP :	model de expansiune ;	— FDEF :	sfîrşit model de expansiune.

Definirea şi utilizarea macroinstrucţiunilor (macro)

În cazul macro se disting, la utilizarea macrogeneratorului, două acţiuni distincte :

— **definirea** care constă în atribuirea unui nume, stabilirea modelului de apel, eventual stabilirea modalităţii de validare a accesului şi definirea modelului de expansiune ;

— **apelul** macroinstrucţiunii, prin citarea modelului de apel cu valori efective pentru eventualii parametrii, din programe autonome sau subprograme (macro sau precompilate).

Sintaxa de definire a unei macroinstrucţiuni este :

: DEFMAC *⟨model de apel⟩*
[: PW *⟨nume program de control⟩*]]/* numai SIRIS-HELIOS */
: EXP
⟨model de expansiune⟩
: FDEF ?

⟨model de apel⟩ : : = *⟨nume macro⟩* [*⟨separator apel⟩*] [*⟨parametrii formali⟩*]
⟨parametrii formali⟩ : : = [*⟨separator apel⟩*] :/: *⟨parametrii formali⟩*
⟨nume macro⟩ : : = este un *⟨identificator⟩* delimitat de separatorii limbajului de manipulare a datelor cu excepţia :

— cuvintelor rezervate ale LDD şi LMD ;
— valorilor numerice ;
— numelor de macroinstrucţiuni, programe precompilate sau programe IMT existente.

Pentru a evita eventualele incoerenţe este indicat ca numele de macroinstrucţiuni să fie diferite de identificatorii caracteristicilor structurii. În cazul în care aceste nume coincid atunci la compilarea structurii, identificatorii vor fi trataţi ca apel al macroinstrucţiunii şi vor fi substituiţi, în locul în care apar, cu modelul de expansiune al acesteia (dacă eventualii parametri sînt corecţi). Mai mult, aceste nume de macroinstrucţiune trebuie să fie unice deoarece, la definirea unei macroinstrucţiuni, sînt clasate în spaţiul dicţionar al bazei de date, iar acţiunea de compilare a programelor şi structurii este subordonată unei căutări, pentru fiecare identificator, citat de utilizator, în acest dicţionar. Existenţa identificatorului în dicţionar declanşează procesul de substituire a acestuia cu modul său de expansiune cu eventuala propagare a valorilor atribuite parametrilor formali în locurile în care aceştia sînt citaţi, după care compilarea continuă cu acest text şi în acest mod.

⟨*separator de apel*⟩ : este reprezentat de unul din separatorii limbajului de manipulare a datelor sau secvențe de identificatori despărțite prin separatorii LMD (maxim 30 caractere) ;

: prezența acestui caracter desemnează poziția unui parametru formal în modelul de apel, parametru utilizabil în modelul de expansiune prin citarea numărului său de ordine în una din următoarele forme :

:n: — parametrul formal va fi înlocuit la apel cu valoarea efectivă a parametrului real (șirul de caractere care formează parametrul real). Acest mod de substituire permite constituirea unor macroinstrucțiuni generalizate, independente de context. Contextul se stabilește la momentul apelului prin citarea, ca valoare a parametrului, unui identificator definit în structura bazei de date ;

':n:' — parametrul formal va fi înlocuit la apel cu constanta alfanumerică care corespunde parametrului real.

La utilizarea parametrilor sînt impuse restricțiile :

— nu este admisă, în cadrul unui program, citarea unui parametru în ambele forme ;

— un parametru nu este pus în factor pentru eventualele macro parametrizate apelate. Dacă macro, pe care o definim, apelează macro parametrizate atunci acestea trebuie să aibă valori reale pentru parametrii ;

— $n \in [1, 16] \cap$ N pentru SOCRATE FELIX și $n \in [1, 30] \cap$ N Mini.

La momentul apelului *n* este înlocuit, acolo unde apare citat în modelul de expansiune, prin valoarea dată parametrului de **rang n** din modelul de apel.

În cazul în care parametrul este citat ca valoare alfanumerică (*':n:'*) valoarea sa efectivă se determină astfel :

— dacă parametrul nu este urmat de un separator de apel : valoarea este dată de șirul de caractere care începe cu primul caracter diferit de spațiu și ține pînă la detectarea primului separator al limbajului de cereri ;

— dacă parametrul este urmat de un separator de apel valoarea sa este dată de șirul de caractere care apare după primul caracter diferit de spațiu pînă la ultimul caracter diferit de spațiu care precede separatorul de apel.

În ambele cazuri, dacă numărul de caractere din parametru este mai mare decît 30 atunci sistemul realizează în mod automat o trunchiere dreaptă a acestuia.

:PW — desemnează sfîrșitul modelului de apel și prezența unui program de control al apelului macroinstrucțiunii de către utilizatori. Această facilitate nu este conservată de implementarea Mini.

⟨*nume program control*⟩ : numele unui program executabil (scris în limbaj de asamblare) care permite o validare suplimentară a parolelor (diferită de cea executată la conexiunea bazei de date) permițînd stabilirea accesului la macroinstrucțiunea respectivă (asigurarea securității datelor).

:EXP — desemnează sfîrșitul programului de control, dacă această facilitate este implementată, sau al modelului de apel și începutul modelului de expansiune ;

:FDEF — desemnează sfîrșitul modelului de expansiune ;

? — lansează în execuție faza de analiză a macroinstrucțiunii și în cazul în care rezultatul acestei analize este corect catalogarea macroinstrucțiunii în spațiul bazei de date (obținerea codului intermediar) ;

⟨*model de expansiune*⟩ :

— declarații sau componente ale declarațiilor exprimate cu ajutorul LDD. Aceste macro vor fi apelate numai sub controlul compilatorului LDD ;

– suită de cereri, o cerere, o parte bine determinată dintr-o cerere, apeluri de subprograme (MAC, PRO sau executabile). Aceste macroinstrucţiuni vor fi apelate sub controlul compilatorului LMD ;

– cuvinte cheie ale LDD sau LMD în cazul în care utilizatorul este familiarizat cu o altă formă de prezentare externă a acestora (utilizatorul îşi defineşte astfel un dicţionar de corespondenţă singurul efect negativ fiind acela al ocupării unui spaţiu suplimentar în baza de date şi al unei creşteri nesemnificative, a timpilor necesari compilării programelor).

Compilarea efectivă a modelului de expansiune se realizează :

– la definirea programelor precompilate pentru macroinstrucţiunile apelate ;

– la lansarea în execuţie a programelor de cereri ;

– la simpla citare a modelului de apel al macroinstrucţiunii sub controlul compilatorului adecvat (compilatorul LMD).

Apelul macroinstrucţiunilor se realizează astfel :

⟨apel macro⟩ :: = *⟨model de apel⟩* / *⟨model de apel⟩⟨apel macro⟩* /
⟨suita de cereri⟩⟨model de apel⟩ /
⟨suita de cereri⟩⟨apel macro⟩

Utilizarea caracterului „?" anunţă compilatorului respectiv terminarea efectivă a unităţii de program destinate analizei şi execuţiei, lansarea efectivă a compilatorului pentru realizarea analizelor sale şi producerea codului de program executabil asociat, iar în cazul corectitudinii acestui cod, lansarea efectivă în execuţie a proramului rezultat.

Sintaxa şi semnificaţia celorlalte ordine ale macrogeneratorului utilizate pentru gestiunea instrucţiunilor este :

1) **suprimarea modelului de expansiune :**

: SUPEXP *⟨nume macro⟩* ?

Textul sursă al macroinstrucţiunii este şters fizic din spaţiul bazei de date, iar la intrarea în dicţionar (pentru *⟨nume macro⟩*) se marchează această operaţie. Intrarea din dicţionar devine disponibilă pentru redefinirea acestui macro (asocierea unui alt model de expansiune recunoscut sub acelaşi nume).

Din experienţa autorilor este indicată utilizarea, pentru catalogarea macroinstrucţiunilor, unor secvenţe de forma :

⟨apel generator⟩
:SUPEXP *⟨nume macro⟩* ?
⟨apel macrogenerator⟩
:DEFMAC *⟨nume macro⟩* ... :FDEF ?

Aceste secvenţe pot fi stocate sub formă de fişiere tip „intrare standard" (proceduri catalogate, de exemplu) în biblioteca sursă asociată (sub)sistemului(lor) informatic(e) pentru care se realizează baza de date. Acest mod de utilizare le face disponibile pe de o parte, pentru manipularea lor cu ajutorul programului bibliotecar iar pe de altă parte pentru utilizarea parametrilor formali ai procedurilor catalogate, ceea ce măreşte puterea de generalizare a acestora. Mai mult, eventualele incoerenţe ale bazei de date (datorate incidentelor hard) detectate pe modelul de expansiune al unei macro pot fi înlăturate utilizînd această secvenţă.

2) **suprimarea unei macroinstrucţiuni :**

:SUPMAC *⟨nume macro⟩* ?

Textul sursă al macroinstrucţiunii este şters fizic din spaţiul bazei de date iar intrarea din dicţionar se blochează (nu este ştearsă ci rămîne indisponibilă, pînă la o eventuală reorganizare a spaţiului dicţionar). În cazul în care se detectează incoerenţe

ale bazei de date pe modelul de expansiune al unei macro iar prin suprimarea şi recatalogarea sa nu dispar (în general nu dispar dacă sînt rupturi de lanţ în înlănţuirea subpaginilor alocate codului sursă al macro) acestea pot fi invalidate prin această suprimare (intrarea din dicţionar nu mai punctează această zonă) şi eventuala recatalogare a programului. Zona din spaţiul bazei de date care nu se recuperează prin suprimare rămîne în continuare incoerentă dar nu afectează în mod real prelucrările. Această zonă va fi recuperată la o eventuală reorganizare a spaţiului real (vezi cap. 6).

3) **editarea unei macroinstrucţiuni :**

:EDIMAC ⟨*nume macro*⟩ ?

După execuţia acestei comenzi textul sursă al macroinstrucţiunii desemnate de ⟨*nume macro*⟩ devine fişier curent şi este pus la dispoziţia editorului de texte SOCRATE pentru eventualele modificări, lansări, în execuţie listări, catalogări etc.

Exemplu :

Realizarea listării codului sursă al unei macroinstrucţiuni se realizează cu secvenţa :

:EDIMAC ⟨*nume macro*⟩ ?
1,$L }comenzi la nivel de linie destinate, editorului de texte
******

4) **listarea numelui şi stării macroinstrucţiunilor stocate în baza de date :**
:LISMAC ?

Listarea se realizează cu linii de forma ⟨stare⟩⟨nume macro⟩ în ordinea definirii temporale a ⟨nume macro⟩.

⟨*stare*⟩ ::! Ъ/*
Ъ (*spaţiu*) : modelul de expansiune asociat este disponibil ;
* : modelul de expansiune a fost suprimat (:SUPEXP) ;
⟨*nume macro*⟩ : numele macroinstrucţiunii.

Observaţie :

În afara parametrilor formali, definiţi în modelul de apel, utilizatorul poate folosi în accepţiunea de parametru variabilele de lucru, poziţionate înaintea apelului la valorile dorite sau poate primi valori cu semnificaţie la execuţia returului apelatului. Evident în aceeaşi accepţiune pot fi utilizate diverse zone de comunicaţie pe care şi le-a rezervat în structura bazei de date sau buffere de memorie structurate conform necesităţilor sale (prin placarea unui *formal logic*).

Exemple de definiri şi apeluri de macroinstrucţiuni :

a) **batch processing :**

⟨apel MACROGENERATOR⟩
:SUPEXP FISA ? /* suprimare, expansiune, dacă există */
⟨apel macrogenerator⟩
:DEFMAC FISA CU CODUL : /* definirea macro */
:EXP I UN PERSOANA AVEC COD = ':1:'; :FDEF ?
⟨apel procesor LMD⟩
FISA CU CODUL 1521001030010 ? /* apelul macro */
:DEFMAC DDN :EXP DE DATA-NASTERII :FDEF ?
⟨*apel macrogenerator*⟩
:SUPEXP CITE ?
⟨*apel macrogenerator*⟩
:DEFMAC CITE PERSOANA S-AU NASCUT IN ANUL: LUNA: ZIUA:
:EXP

```
M Y1 = D TOUT PERSOANA AYANT
                          AN   DDN    = :1:
                       ET LUNA DDN    = :2:
                       ET   ZI DDN    = :3:;
I(1) 'NUMARUL PERSOANELOR DIN B.D.'
I (+1) 'NASCUT LA ACEASTA DATA ESTE:' I(+6 -5) Y1 ECRIRE
:FDEF ?
⟨apel procesor LMD⟩
CITE PERSOANE S-AU NASCUT IN ANUL 1960 LUNA 10 ZIUA 3 ?
⟨apel macrogenerator⟩
:SUPEXP FUN ?
⟨apel macrogenerator⟩
:DEFMAC FUN:; EXP UN FUNCTII AVEC COD = :1:; :FDEF ?
⟨apel macrogenerator⟩
:DEFMAC IF :EXP SI :FDEF ?
⟨apel macrogenerator⟩
:DEFMAC THEN :EXP ALORS :FDEF ?
⟨apel macrogenerator⟩
:DEFMAC ELSE :EXP SINON :FDEF ?
⟨apel macrogenerator⟩
:DEFMAC ENDIF; :EXP FIN :FDEF ?
⟨apel macrogenerator⟩
:DEFMAC NEXT SENTENCE :EXP /* SECVENTA VIDA */:FDEF ?
⟨apel macrogenerator⟩
:SUPEXP ACTNOM ?
⟨apel macrogenerator⟩
:DEFMAC ACTNOM : UZILIZIND CODUL : :EXP
/* ACEST PROGRAM REALIZEAZA CREEREA, MODIFICAREA SAU STERGEREA
  REALIZARILOR
/* ENTITATII NOMENCLATOR :1: UTILIZIND UN COD DE TIPUL :2: (NUMERIC-
  CODNUM;
/* ALFANUMERIC — CODALF)                                        **//
D X1 D X2
FAIRE M :2: DE X0 = U M :2: DE X0 = EXT
  IF :2: DE X0 = U        /* IF = SI */
   THEN                   /* THEN = ALORS */
     I (1) '**ERR.01.COD ... VID ?' ECRIRE
     REFAIRE
   ENDIF;                 /* ENDIF; ; FIN */
M X1 = UN :1: AVEC COD = :2:;
SI PAS X1
  ALORS
     G UN :1: X2
     M COD DE X2 = :2: DE X0
     SI COD DE X2 = U
       ALORS
         I (1) '**ERR.02.COD ...VALOARE' I (+1) 'ERONATA ?' ECRIRE
         S X2 REFAIRE
     ENDIF;
FIN
M X1 = U M Y1 = 0
M X1 = UN :1: AVEC COD = :2: DE X0;
FAIRE
   IF Y1 > 3
    THEN
       I(1) '**ERR.03.⟨ACTNOM⟩...' I(+1) 'ABANDONAT !'
       I(+1) 'EROARE DE OPERARE ?' ECRIRE
       SORTIE
FIN
SI EXISTE DEN1 DE X1 ALORS I DEN1 DE X1 FIN
M DEN1 DE X1 = EXT
M Y1 = Y1 + 1
```

```
IF DEN1 DE X1 ¬= U
  THEN
    IF DEN1 DE X1 ='STERS'
    THEN
      I(1) '**EXECUT STERGEREA ? (DA,NU): '
      M RASPUNS DE X0 = U M RASPUNS DE X0 = EXT
      SI RASPUNS DE X0 = 'DA'
        ALORS
          POUR X1
            M COD = U M DEN1 = U M DEN2 = U
          FIN
          S X1
        FIN
        SORTIE
      ELSE
          IF DEN2 DE X1 ¬ = U THEN I DEN2 DE X1 ENDIF;
          M DEN2 DE X1 = EXT
          SORTIE
        ENDIF;
      ELSE
          I(1) '**ERR.01.DEN1 ... VID ?' ECRIRE
          REFAIRE
        ENDIF;
    FIN       I' '
      M CONTINUATI = U M CONTINUATI = EXT
      SI CONTINUATI = 'DA' ALORS REFAIRE FIN
FIN
D X1  D X2
:FDEF ?
```

⟨*apel procesor* LMD⟩

```
ACTNOM SANCTIUNE UTILIZIND CODNUM ? /* entitatea SANCTIUNE */
```

01 MUSTRARE- SCRISA NU	COD DEN1 DEN2 CONTINUATI	simularea dialogului purtat cu aceste obiecte cu ajutorul cartelelor perforate

⟨*apel procesor* LMD⟩

```
ACTNOM FUNCTII UTILIZIND CODNUM ? /* entitatea FUNCTII */
```

⟨*apel procesor* LMD⟩ —

```
ACTNOM SPECIALITATI UTILIZIND CODNUM ? /* entitatea SPECIALITATI */
```

⟨*apel procesor* LMD⟩

```
ACTNUM LOCALITATI UTILIZIND CODNUM ? /* entitatea LOCALITATI */
```

⟨*apel procesor* LMD⟩

```
ACTNOM PROFESII UTILIZIND CODALF ? /* entitatea PROFESII */
```

⟨*apel macrogenerator*⟩

```
:DEFMAC ID-PERS :EXP
 POUR X1
  I(1) NUME I (+1) PRENUME
  I(+2) DEN1 DE PERS-PRO ECRIRE
FIN :FDEF ?
```

b) conversaţional :

Apelul procesorului macrogenerator se realizează cu comanda *GO M*.

```
$GO M ⟨cr⟩ /* definire macroinstructiune */
QUESTION :DEFMAC IDENTIFICA PERSOANA :EXP ⟨cr⟩
D X1 FAIRE ⟨cr⟩
I 'INTRODUCETI CODUL PERSOANEI' ⟨cr⟩
M CODALF DE X0 = U M CODALF DE X0 = EXT ⟨cr⟩
SI EXISTE UN PERSOANA X1 AVEC COD = CODALF DE X0; ⟨cr⟩
  ALORS ⟨cr⟩)
    ID-PERS ⟨cr⟩
```

```
   SINON ⟨cr⟩
      I '* NU EXISTA ACEASTA PERSOANA ?' ⟨cr⟩
FIN ⟨cr⟩
M CONTINUATI = U M CONTINUATI = EXT ⟨cr⟩
SI CONTINUATI = 'DA' ALORS REFAIRE FIN ⟨cr⟩
FIN :FDEF ? ⟨cr⟩
QUESTION : NON ⟨cr⟩
$ GO ⟨cr⟩
   IDENTIFICA PERSOANA ? /* apel */
[dialogul cu utilizatorul]
QUESTION: ⟨cr⟩
ACTNOM FUNCTII UTILIZIND CODNUM ?
:  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
QUESTION: ⟨cr⟩
:DEFMAC HELP-NOM :EXP ⟨cr⟩
I(1) '* ASOCIERILE DINTRE MNEMONICE ' ⟨cr⟩
I(+1) 'SI NOMENCLATOARELE UTILIZATE IN SISTEM' ⟨cr⟩
I(+1) 'SINT:' ECRIRE ⟨cr⟩
I(2) '=SAN : SANCTIUNE;' ECRIRE  ⟨cr⟩
I(2) '=FUN :FUNCTII; ECRIRE ⟨cr⟩
I(2) '=SPE : SPECIALITATI; 'ECRIRE ⟨cr⟩
I(2) '=LOC : LOCALITATI;' ECRIRE ⟨cr⟩
I(2) '=PRO : PROFESII;' ECRIRE ⟨cr⟩
:FDEF ? ⟨cr⟩
QUESTION::DEFMAC DORESC SA LUCREZ CU : ⟨cr⟩
           :EXP EXEC :1:  :FDEF ?
```

Programe precompilate

Un program precompilat reprezintă o unitate sintactică de program de cereri direct executabilă (la simplul apel) şi, eventual, adaptabilă la un context specific de execuţie (în cazul în care este construit independent de context).

Faţă de o macroinstrucţiune un program precompilat nu utilizează nici parametri formali şi nici separatorii de apel. Eventualii parametrii de execuţie pot fi furnizaţi la momentul apelului, prin :

— variabilele de adresă X_i (pentru eventuala stabilire a contextului de execuţie) ;

— variabilele de lucru Y_i, Z_i şi W_i (care pot conţine atît valori destinate unor prelucrări specifice cît şi valori care vizează selecţia logicii de execuţie a modelului de expansiune) ;

— buffere de lucru (rezervate, structurate şi completate, eventual, cu valori).

Returul parametrilor către apelant se efectuează în aceeaşi manieră. Programul poate fi constituit ca un program independent sau ca subrutină. Un program precompilat poate apela programe precompilate, macro şi programe executabile şi poate fi apelat de alte programe precompilate, de macroinstrucţiuni, de programe de cereri sau de programe scrise într-un limbaj de nivel înalt, prin intermediul „interfeţei cu limbaje evoluate".

Sintaxa de definire a unui program precompilat este :

:DEFPRO ⟨*nume program*⟩
  [:PW ⟨*nume program control* ⟩]
[:CONTXT {X0 [⟨*citaţie formală entitate*$_i$⟩]

[⟨*citaţie formală*$_i$⟩]
[D X_1 = ⟨*citaţie formală entitate*$_1$⟩]
[D X_1 = *citaţie formatare buffer*$_1$⟩]}]
:EXP
⟨*model de expansiune*⟩
:FDEF ?
⟨*nume program*⟩ : numele programului (idem ⟨*nume macro*⟩) ;
:PW – verificarea accesului la acest program se va efectua, la orice tentativă de apel, prin intermediul programului IMT specificat de ⟨*nume program control*⟩
⟨*citaţie formală entitate*⟩ ::= UN ⟨*nume entitate*⟩ [⟨*calificare*⟩].

Toţi indicatorii aflaţi în partea ⟨*stîngă*⟩ a comenzilor LMD, prezenţi în modelul de expansiune, care nu au precizată o calificare explicită sau nu sînt calificaţi implicit, prin punerea în factor a unui calificator, vor fi consideraţi ca elemente definite în ⟨*nume entitate*⟩ (deci calificaţi implicit prin ⟨*nume entitate*⟩)
⟨*citaţie formală*$_i$⟩ ::= D X0 / D X_i, $i = \overline{1,9}$
⟨*citaţie formatare buffer*$_1$⟩ ::= FORMAL ⟨*nume formal*⟩ DANS ⟨*nume buffer*⟩ ⟨*număr buffer*⟩

Ordinea de efectuare a declaraţiilor de formatare a bufferelor trebuie să fie identică cu cea declarată în programul apelant.

Utilizatorul poate solicita rezervarea şi formatarea bufferelor în cadrul modelului de expansiune cu ajutorul comenzii M (respectînd regula precedentă).
:EXP – desemnează începutul modelului de expansiune ;
⟨*model de expansiune*⟩ : unitate de program de cereri executabilă ;
:FDEF – sfîrşitul modelului de expansiune ;
? – lansează în execuţie macrogeneratorul pentru obţinerea codului compilat direct executabil. Dacă nu este detectată nici-o eroare, în cadrul acestei faze, atunci :

– codul sursă al programului va fi stocat în spaţiul „FICH" al bazei de date (este ţinut în această formă pentru a putea fi recuperat şi, eventual, modificat cu editorul de texte) ;

– codul compilat al programului va fi stocat în spaţiul PROG al bazei de date (destinat apelurilor în execuţie);

:CONTXT – stabileşte contextul în care este apelat programul, şi/sau modul de utilizare a variabilelor de tip X_1 drept calificatori, conform regulilor (R) următoare :

R1 : pentru ca un program să poată fi apelat în orice context este necesar :
– să fie definit cu cîmpul :CONTXT urmat în mod unic de eventualele definiri ale variabilelor X_1 ;
– să fie independent de contextul care îl înglobează adică toate citaţiile externe sau aflate în stînga să fie terminate prin „DE X_1" ;

R2 : pentru ca un program să poată fi apelat în contextul ⟨FICHIER⟩ trebuie să se definească acest lucru :
– în cîmpul : CONTXT prin D X0 ;
– în cîmpul : CONTXT X0.

R3 : pentru ca un program să poată fi apelat în contextul unei anumite entităţi trebuie să fie definit astfel :
– fie cu cîmpul : CONTXT ⟨*citaţie formală entitate*⟩ ;
– fie cu cîmpul : CONTXT fără citaţie formală de context dacă programul este independent de contextul care-l înglobează ;

R4 : deoarece variabilele de tip X_i sînt statice, utilizarea lor în cadrul programelor precompilate, trebuie să respecte următoarele reguli :

– numai variabilele X_i care sînt utilizate ca pronume de apel : ...DE X_i din modelul de expansiune trebuie să fie definite în mod obligatoriu în cîmpul : CONTXT prin D X_i.

Această definire formală nu antrenează punerea la nedefinit a desemnării reale ci din contră reprezintă modalitatea de transmitere a valorii variabilei X_i de la apelant la programul apelat, la execuția apelului : variabilele X_i, definite în cîmpul :CONTXT, reprezintă deci parametrii de intrare.

Exemplu :

```
⟨definire⟩                                          ⟨apel⟩
:DEFPRO FISA                                    M Z1 ='1521001030010'
:CONTXT D X1 ←  stabilire valoare  →  M X1=UN PERSOANA AVEC COD = Z1;
:EXP POUR X1    parametru de apel     EXEC FISA /* apel program */

  I(1) NUME I(+1) PRENUME                  ?
  ECRIRE
  FIN
:FDEF ?
```

– toate variabilele X_i care nu sînt definite cu cîmpul ;CONTXT sînt considerate ca disponibile, adică pot fi definite formal, în modelul de expansiune, fără restricție.

Regulile de valabilitate contextuală nu funcționează între apelant și apelat ci numai în interiorul programului apelat ;

– toate variabilele X_i utilizate într-un program (cu definire sau nu în cîmpul :CONTXT) sînt retransmise apelantului la sfîrșitul execuției programului apelat deci reprezintă parametrii de ieșire. Acest lucru implică faptul ca desemnările formale să fie identice :

– în apelant ;

– la ieșirea (și numai la ieșirea) programului apelat.

Deoarece regulile de valabilitate contextuală funcționează în interiorul programului este necesar ca definirile formale ale variabilelor X_i, la ieșirea din program, să fie făcute la cel mai înalt nivel.

Sintaxa de apel a programelor precompilate este :

EXEC ⟨*nume program*⟩

La apel are loc verificarea identității desemnărilor formale ale variabilelor X_i între apelant și utilizarea lor la execuția programului apelat. Eventualele erori de utilizare a variabilelor X_i sînt semnalate prin mesaje de eroare.

Sintaxa și semnificația celorlalte ordine ale macrogeneratorului utilizate pentru gestiunea programelor precompilate este :

1) suprimarea modelului de expansiune :

:SUPEXP ⟨*nume program*⟩ ?

Acțiunea ordinului se desfășoară astfel :

– se șterge codul compilat (asociat) din spațiul „PROG" ;

– se șterge codul sursă asociat din spațiul bazei de date ;

– se marchează la intrarea din dicționar, asociată pentru ⟨*nume program*⟩, starea programului.

Pentru (re)catalogarea unui program precompilat se va utiliza secvența :

⟨*apel macrogenerator*⟩

:SUPEXP ⟨*nume program*⟩ ?

⟨*apel macrogenerator*⟩
:DEFPRO ⟨*nume program*⟩ ... :FDEF ?
În cazul redefinirii unui subprogram cu un context de execuţie diferit de contextul precedentei definiri este necesară recompilarea tuturor programelor apelante.

2) **suprimarea unui program precompilat :**
:SUPPRO ⟨*nume program*⟩ ?
Diferenţa între :SUPEXP şi :SUPPRO este aceea că :SUPPRO realizează o blocare a intrării din dicţionar asociate numelui programului.

3) **editarea unui program precompilat :**
:EDIPRO ⟨*nume program*⟩ ?
Textul sursă asociat lui ⟨*nume program*⟩ este recuperat din spaţiul bazei de date şi este pus la dispoziţia ordinelor editorului de texte (devine „fişier curent").

4) **listarea numelui şi a stării programelor precompilate :**
:LISPRO ? (idem ⟨*macro*⟩)

Exemplu :

Vom defini un program precompilat care realizează o „monitorizare" a programelor, construite anterior, pentru manipularea realizărilor nomenclatoarelor utilizate de sistem :

```
⟨apel macrogenerator⟩
:DEFPRO NOMENCLATOARELE :CONTXT X0
:EXP M Y7 = 0
 FAIRE D  X1  D X2  FIN
 FAIRE
   I(1) 'ACEST PROGRAM PERMITE'
   I(+1) 'CREEREA,ACTUALIZAREA,STERGEREA'
   I(+1) 'SI INTEROGAREA REALIZARILOR'
   I(+1) 'NOMENCLATOARELOR, ' ECRIRE I''
   I(2)  'SELECTIND NOMENCLATORUL DORIT'
   I(+1) 'CU MNEMONICA ASOCIATA. 'ECRIRE I''
   I(2)  'DACA NU CUNOASTETI ASOCIERILE DINTRE'
   I(+1) 'MNEMONICELE SI NOMENCLATOARE TASTATI HELP .' ECRIRE I''
   I(2)  'DACA DORITI SA TERMINATI PRELUCRAREA'
   I(+1) 'TASTATI END .' ECRIRE
   I 'SELECTATI NOMENCLATORUL DORIT :'
   M MNEMONICA = U M MNEMONICA = EXT
   SI MNEMONICA = U
     ALORS
        M Y7 = Y7 + 1
        SI Y7 > 5   ALORS REFAIRE FIN
        I (1)   '** NU CREDETI CA TREBUIE'
        I(+1) 'SA INVATATI SA OPERATI !' ECRIRE
        I(1)   'ABANDON LA 6 ERORI DE OPERARE'
        I(+1) 'CONSECUTIVE !' ECRIRE
        SORTIE
   FIN M Y7 = 0
   SI MNEMONICA = 'END'
        ALORS
          I '*PRELUCRARE TERMINATA NORMAL *'
          I 'LA REVEDERE I'
          SORTIE
     FIN
     SI MNEMONICA = 'HELP'
        ALORS
           HELP-NOM REFAIRE
        FIN
```

```
        M Y8 = 0
        SI MNEMONICA = 'SAN'
          ALORS
        ACTNOM SANCTIUNE UTILIZIND CODNUM
        M Y8 = 1
        FIN
        SI MNEMONICA = 'FUN'
          ALORS
            ACTNOM FUNCTII UTILIZIND CODNUM
            M Y8 = 1
        FIN
        SI MNEMINICA = 'SPE'
          ALORS
            ACTNOM SPECIALITATI UTILIZIND CODNUM
            M Y8 = 1
        FIN
          SI MNEMONICA = 'LOC'
        ALORS
            ACTNOM LOCALITATI UTILIZIND CODNUM
            M Y8 = 1
        FIN
          SI MNEMONICA = 'PRO'
          ALORS
            ACTNOM PROFESIA UTILIZIND CODALF
            M Y8 = 1
        FIN
        SI Y8 ┐= 1
          ALORS
            I(1)   '*** ÎMI PARE RĂU DAR'
            I(+1) 'NU GESTIONEZ NOMENCLATORUL'
            I(+1) 'CU MNEMONICA' I(+1) MNEMONICA
            ECRIRE
        FIN
        M CONTINUATI = U M CONTINUATI = EXT
        SI CONTINUATI = 'DA' ALORS REFAIRE FIN
      FIN
      D X1  D X2
      :FDEF ?
```

Apelul acestui program se realizează astfel:

1) EXEC NOMENCLATOARELE ?
2) prin intermediul macro „DORESC“:
DORESC SA LUCREZ CU NOMENCLATOARELE ?
(o frază lizibilă şi înţeleasă de orice tip de utilizator !?).

Programe executabile (SIRIS/HELIOS)

Acest tip de programe permit utilizatorului să descrie anumite funcţii necesare aplicaţiilor sale, imposibil de realizat (sau ineficiente) cu ajutorul comenzilor LMD. Programele sînt scrise în limbaj de asamblare (ASSIRIS), iar schimbul de parametrii se realizează respectînd reguli specifice fiecărui tip.

Utilizatorul poate construi trei tipuri de programe, numite IMT, conform sistemului de operare gazdă, astfel:

1) — **programe de „hashing“ (dispersare) : aceste programe sînt asociate unor carac**teristici declarate chei de acces în structura bazei de date şi sînt apelate în

mod automat la execuţia unei comenzi care solicită acces la dicţionarul asociat caracteristicii ;

2) — **programe de control a drepturilor de acces** : sînt apelate în mod automat la orice tentativă de lansare în execuţie a macroinstrucţiunilor şi programelor precompilate, cărora li se asociază (:PW...) ;

3) — **programe la dispoziţia utilizatorului** apelabile prin comanda EXEC ⟨*nume program IMT*⟩.

Sintaxa de definire

: DEFIMT ⟨*nume program IMT*⟩ ?

⟨*linii IMT*⟩

: DEFIMT : început definire program IMT ;

⟨*nume program IMT*⟩ : nume atribut în antetul fişierului IMT rezultat după link-editarea programului ASSIRIS cu ordinul .LINK FN : ⟨*nume program IMT*⟩, PUN.

Acest nume se formează conform regulilor de definire a numelor de programe (max. 8 caractere alfanumerice).

⟨*linii IMT*⟩ : linii rezultate la dispozitivul de tip „punch" al sistemului de calcul (fişier tip *3) ca urmare a prezenţei argumentului PUN în comanda de lansare a editorului de legături.

Versiunea SOCBTCH V.1.5 a C.Ce. al C.S.P., admite definirea codului IMT al programului dintr-o bibliotecă IMT a utilizatorului astfel :

:DEFIMT ⟨*nume program*⟩
DV:ADxx, GN:g,VN:v,FN: '⟨*nume bibliotecă*⟩ IMT/⟨*nume program*⟩'UN:*u*
unde :

— ADxx : suportul bibliotecii ;
— *g* : numărul de generare al bibliotecii ;
— *v* : numărul de versiune al bibliotecii ;
— ⟨*nume bibliotecă*⟩ : numele bibliotecii pe 8 caractere (eventual completat cu spaţii la dreapta pînă la această lungime) ;
— IMT : biblioteca trebuie să fie de tip IMT ;
⟨nume program⟩ : numele programului dorit ;
— *u* : varianta de program (*u* = 255 cel mai recent program)

Comunicarea între programele IMT şi programele SOCRATE ale utilizatorului se realizează cu ajutorul unei zone de memorie continuă, descrisă în programele scrise în limbajul de asamblare ca secţiune fictivă (DSECT) şi accesibilă prin registrul R11, considerat registru de bază.

Această zonă comună conţine valorile variabilelor Y_1 şi Z_1 şi a parolei de acces a utilizatorului PW.

Structura acestei zone este (V.1.5) :

```
ZONCOM      DSECT
            ORG    X'OEO'  (RES,4 56)
  Y1        RES,4   1
  Y2        RES,4   1
  .           .
  .           .
  .           .
  Y25       RES,4   1
            ORG    X'144'  (RES,4 8)
  Z1        RES,4   8
  Z2        RES,4   8
  .           .
  .           .
```

```
.                    .
.                    .
Z7                   RES,4   8
PW                   EQU     324
```

şi devine accesibilă programului prin declaraţia:

USD, ZONCOM 11.

Returul, din subprogram la SOCRATE se realizează prin instrucţiunea BRU 0,8.

Un program IMT trebuie să respecte următoarele cerinţe:

— să nu fie segmentat;
— să nu utilizeze comenzi SGF;
— să fie reentrant;
— să fie paginat:
— să ocupe maximum 64 de pagini (1 pagină = 255 cuvinte);
— să înceapă cu o instrucţiune executabilă;
— să nu utilizeze literali;
— să nu modifice valorile următoarelor registre: R7, R11, R12, R13, R14 şi R15 (în cazul în care se doreşte modificarea trebuiesc salvate iar la retur trebuie restabilit conţinutul iniţial).

Modul de utilizare a acestor registre este următorul:

— R7: efectuarea cererilor de rezervare şi eliberare a zonelor de lucru;
— R11: registru de bază pentru acces la Y_i, Z_i, W_i şi PW;
— R12: registru de bază al zonei de lucru;
— R13: efectuarea de salturi interpretate;
— R14: registru de bază al programului;
— R15: registru de bază al tabelei de segmente.

Pentru a permite utilizatorului alocarea dinamică a unei zone temporare de lucru s-a definit comanda specială:

EMPI COM, 32, 16, 8,8 X'8639001C',X'2003',AF(1),X'00'

utilizabilă prin:

[⟨*eticheta*⟩] EMPI ⟨*număr cuvinte*⟩, unde:

⟨*număr cuvinte*⟩ : = numărul de cuvinte rezervate (≤ 512).

O zonă temporară de lucru este un buffer de lucru temporar alocat în exclusivitate unui utilizator.

Aceste zone se eliberează, după utilizare, cu ajutorul comenzii speciale:

DEPI COM, 32, 16, 8,8, X'8639001C', X'2303',AF(1), X'00'

executabilă prin ordinul:

[⟨*eticheta*⟩] DEPI ⟨*număr cuvinte*⟩

Într-un program se pot succede mai multe cereri de rezervare, ultima cerere, fiind cea accesibilă utilizatorului.

Cererile de eliberare trebuie să se facă în sens invers celor de alocare (⟨*număr cuvinte*⟩ din cererea de eliberare trebuie să fie egal cu ⟨*număr cuvinte*⟩ din cererea de alocare corespondentă) şi trebuie comandate obligatoriu înaintea returului la SOCRATE.

Zona temporară definită este bazată prin registrul R12.

În cazul în care mărimea unui program depăşeşte 255 cuvinte (1 pagină) este posibilă segmentarea lui prin apel la o instrucţiune specială de salt interpretat:

[⟨*eticheta*⟩] BRIN ⟨*p*⟩, ⟨*d*⟩ unde:

⟨*p*⟩ : numărul paginilor ($1 \leq p \leq 64$)

⟨*d*⟩ : deplasarea relativă (în octeţi) în pagină ($0 \leq d \leq 1020$).

Instrucţiunea trebuie să fie definită, înaintea utilizării, prin comanda specială:

BRIN COM,32,16,16, X'D6390000',AF(1),AF(2).

Sintaxa şi modul de utilizare a celorlalte ordine ale macrogeneratorului utilizate pentru gestiunea programelor IMT :

1) **suprimarea modelului de expansiune :**

: SUPEXP ⟨*nume program*⟩ ?

Modelul de expansiune este şters fizic din spaţiul „PROG" al bazei de date şi se marchează starea programului la intrarea asociată numelui său în dicţionar.

La catalogarea programelor IMT este indicată utilizarea unei secvenţe de forma :

⟨apel macrogenerator⟩
:SUPEXP ⟨nume program⟩ ?
⟨apel macrogenerator⟩
:DEFIMT ⟨nume program⟩ ?
. . . .

2) **suprimarea unui program IMT :**

: SUPIMT ⟨*nume program*⟩ ?

Textul programului este şters fizic din spaţiul „PROG" al bazei de date iar intrarea din dicţionar asociată numelui este blocată.

3) **listarea numelui şi a stării programelor IMT stocate în baza de date :**

: LISIMT ?

Programe de control a drepturilor de acces

Administratorul bazei de date declară utilizatorii bazei de date prin specificarea elementelor de identificare (*nume*, *număr*) şi autentificare (*parola*). În afara acestor elemente administratorul asociază utilizatorilor anumite drepturi de acces care se referă la permiterea sau interzicerea efectuării anumitor operaţii cu obiectele gestionate de baza de date. Aceste drepturi de acces se stochează pe un cuvînt binar (blocul DROIT-UTILISATEUR din entitatea UTILISATEUR) cu structura (pentru mai multe informaţii vezi cap. 6) :

⟨acces macro⟩	⟨acces limbaj⟩	⟨parola de acces la operaţii⟩ (1 ÷ 16777215)			(PW)
octetul 0		octetul 1	octetul 2	octetul 3	

Sistemul SOCRATE efectuează un control automat la LOGIN, pe datele de identificare şi autentificare şi pe primul octet (0) al acestui cuvînt. Rolul principal al programelor de control este acela de a testa ⟨*parola de acces la operaţii*⟩ pentru a asigura o discreţie suplimentară a datelor din baza de date (această testare poate fi combinată, cu o testare şi a octetului 0, pentru selectarea unei anumite combinaţii a acestuia). Aceste programe se construiesc de către administratorul bazei de date (care, de altfel, acordă sau ridică drepturile de acces ale utilizatorului) şi pot fi schimbate (din punct de vedere al conţinutului, nu şi numele) fără repercusiuni asupra programelor care le apelează.

Acest cuvînt este pus la dispoziţia utilizatorului prin instrucţiunea LD4,R 11.PW (R reprezintă unul din registrii generali cu care utilizatorul are dreptul să lucreze). Dacă un utilizator nu îndeplineşte condiţiile impuse de program atunci

anularea dreptului său de acces la macro sau programul precompilat apelat se realizează prin secvenţa :

```
LD4I,R  0
ST4,R   11.PW
```

Exemplu :

Programul DISCRET asigură accesul la macroinstrucţiunile sau programele precompilate care îl citează în cîmpul : PW numai utilizatorilor care îndeplinesc condiţiile :

– ⟨*acces macro*⟩ : : = IMT + COMP + MAC (7) ;
– ⟨*acces limbaj*⟩ : : = CREATION (3) ;
– ⟨*parola de acces la operaţii*⟩ este 1317215.

Structura programului este :

```
 1. ZONCOM  DSECT
 2. PW      EQU    324
 3. DISCRET CSECT
 4.         LD4,0  11.PW
 5.         CP1I,0 X'73'      ⟨acces macro⟩⟨acces limbaj⟩ = X'73'
 6.         BNZ    FARAACC    NU → ANULEAZA DREPTURILE
 7.         LD1I,0 0          DA → IZOLEAZĂ PAROLA (NUMARUL)
 8.         LD4I,1 1317215
 9.         CP4,0  4          ARE NUMARUL DORIT ?
10.         BZ     RETUR      DA
11. FARAACC RES,4  0
12.         LD4I,0 0
13.         ST4,0  11.PW      ANULEAZA DREPTUL DE ACCES
14. RETUR   RES,4  0
15.         BRU    0,8        RETUR LA APELANT (SOCRATE)
16.         END    DISCRET
```

Dacă administratorul doreşte să schimbe acest program pentru a acorda drept de acces tuturor utilizatorilor care au parola de acces la operaţii în intervalul [101, 251] atunci trebuie să execute operaţiile :

– se înlocuiesc liniile 8, 9 şi 10 cu secvenţa :

```
1. LD4I,1   101
2. CP4,0    4
3. BLZ      FARAACC    <101
4. LD4I,1   251
5. CP4,0    4
6  BGZ      FARAACC    >251
```

– se compilează şi linkage-editează programul transformat cu acelaşi nume DISCRET ;

– se suprimă modelul de expansiune al programului cu ordinul : SUPEXP DISCRET ! ;

– se recatalogheză ultima versiune a programului.

Programe IMT pentru determinarea unei intrări în dicţionar (hashing)

Programele IMT pentru hashing (randomizare, dispersare), asociază unei caracteristici (valoarea numerică sau cuvînt) un cod care dă adresa de intrare într-un dicţionar. Valoarea rezultată în urma aplicării funcţiei de hash-code trebuie să fie cuprinsă între 0 şi mărimea dicţionarului (max. 255 ; vezi ⟨*index*⟩ cap. 3).

Aceste programe sînt apelate în mod automat iar schimbul parametrilor se desfăşoară astfel (R_i desemnează registrul general i, al sistemului de calcul):

1) Parametrii de intrare:

– R 2: conţine mărimea dicţionarului;

– R 3: conţine:

• adresa şirului alfanumeric, pentru caracteristici de tip cuvînt;

• valoarea numerică, pentru valoare numerică.

Un şir alfanumeric se memorează ca o caracteristică ASSIRIS declarată cu TEXTC:

l	caractere codificate în EBCDIC

l = lungime lanţ caractere ($1 \leqslant l \leqslant 255$)

2) Parametrii de ieşire:

– R 1: conţine valoarea hash-code;

– R 2: conţine un reprezentant al valorii sau cuvîntului căruia i s-a calculat codul hashing (valoarea se va reprezenta pe ultimii trei octeţi).

Această valoare este destinată ameliorării performanţelor sistemului şi funcţiei de randomizare în sensul rezolvării coliziunilor (numărul de intrări în dicţionar este redus comparativ cu numărul valorilor cărora li se aplică funcţia de randomizare, deci implicit numărul de sinonime va fi destul de mare).

Programe la dispoziţia utilizatorului

Aceste programe sînt apelate de utilizator din programele SOCRATE cu ajutorul ordinului EXEC ⟨*nume program* IMT⟩. Schimbul de parametri între apelat şi apelant se realizează prin intermediul zonei de comunicaţie (ZONCOM). Valorile atribuite parametrilor de intrare pot constitui elemente de selecţie a funcţiunilor realizate de subprogram, alegerea lor fiind la dispoziţia utilizatorului (alegerea se execută în limita specificaţiilor de utilizare a fiecărui program în parte).

Exemplu:

Dorim să realizăm, în cazul lucrului cu variabilele Z_i – funcţia de inserare a unui şir de caractere dintr-o variabilă Z_i, în altă variabilă fără a distruge restul conţinutului acesteia (LMD oferă numai posibilitatea concatenării şirurilor de caractere considerînd drept receptor toate poziţiile disponibile în dreapta celei specificate ca origine a concatenării). Pentru a realiza acest lucru vom defini o macro INSERT care are rolul de a valida parametrii acţiunii şi a pregăti parametrii de apel ai unor subprograme IMT. Sintaxa de apel a acestei macro respectă sintaxa de utilizare a variabilelor de tip Zi la concatenare astfel:

INSERT ⟨Z_r⟩ (⟨*poziţie început*$_1$⟩ ⟨*lungime*$_1$⟩) =
⟨Z_i⟩ (⟨*poziţie început*$_2$⟩ ⟨*lungime*$_2$⟩)

Singura diferenţă este faptul că atribuirea nu se mai face cu comanda M şi toţi parametrii sînt obligatorii (indiferent care este valoarea lor).

⟨Z_r⟩: variabila de tip Z_i desemnată ca receptor, r = 1,6;

⟨Z_e⟩: variabila de tip Z_i desemnată ca emitor, e = 1,6;

⟨*poziţie început*$_1$⟩: număr întreg specificînd poziţia din variabila ⟨Z_r⟩ din care începe inserarea;

$\langle \mathit{lungime}_1 \rangle$: lungimea pe care se va efectua inserarea ;
$\langle \mathit{poziţie\ început}_2 \rangle$: număr întreg specificînd poziţia din variabila $\langle Z_e \rangle$ de unde începe şirul de inserat ;
$\langle \mathit{lungime}_2 \rangle$: lungimea şirului de inserat.

Aceşti parametrii care definesc poziţiile şi lungimile trebuie să îndeplinească condiţiile :

– $[\langle \mathit{poziţie\ început}_1 \rangle + \langle \mathit{lungime}_1 \rangle] \in [1, 30]$;
– $[\langle \mathit{poziţie\ început}_2 \rangle + \langle \mathit{lungime}_2 \rangle] \in [1, 30]$.

Convenţiile de funcţionare a acestei macro sînt :

– dacă $\langle \mathit{lungime}_2 \rangle$ este mai mare decît $\langle \mathit{lungime}_1 \rangle$ şirul emiţător va fi trunchiat la dreapta cu marcarea unui mesaj de atenţionare ;
– dacă $\langle \mathit{lungime}_2 \rangle$ este mai mică decît $\langle \mathit{lungime}_1 \rangle$ şirul receptor va fi completat, pe lungimea rămasă, cu spaţii la dreapta ;
– controlul variabilei $\langle Z_r \rangle$ va fi făcut egal cu numărul de caractere definite (poziţia ultimului caracter definit) ;
– variabila Y24 va fi utilizată ca manevră ;
– variabila Z7 va fi utilizată pentru a forma o „mască" a legii de inserare. La returul la apelant al programului IMT dacă octetul 1 al acestei variabile este diferit de spaţiu atunci operaţia s-a efectuat eronat.

Dăm în continuare codul sursă al macro INSERT şi al programelor IMT apelate :

```
:DEFMAC INSERT : (: :) = :(: :)
:EXP
M Z7 = U /* anulare variabilă de lucru */
FAIRE I(1) '**INSERT**ERR.' /* antet mesaj eroare */
  M Z7 = ':1:'
  SI Z7 ⟨'Z1' OU Z7⟩'Z6'
    ALORS
      I(+0) '01. ⟨RECEPTOR⟩ # ZI ?' ECRIRE SORTIE
FIN
M Z7 = U M Z7 = ':4:'
SI Z7 ⟨'Z1' OU Z7⟩ 'Z6'
  ALORS
    I(+0) '02. ⟨EMITOR⟩ +ZI ? 'ECRIRE SORTIE
FIN M Z7 = U /* FORMAREA LEGII DE INSERARE */
M Z7 (2) = ':1:' M Y 24 = Z7 (3)
EXEC LEXINSRT
M Z7 (3) = ':4:' M Y24 = Z7 (4)
EXEC LEXINSRT
M Y24 = :2:
SI Y24 = U
  ALORS
    I(+0) '03.VALOARE NUNUMERICA ?' ECRIRE SORTIE
FIN
EXEC LEXINSRT
M Y24 = :3:
SI Y24 = U
  ALORS
    I(+0) '03.VALOARE NENUMERICA ? ECRIRE SORTIE
FIN
EXEC LEXINSRT
M Y24 = Y24 + :2:
SI Y24 ⟨ = 0 OU Y24 ⟩ 31
  ALORS
    I(+0) '04.⟨POZITIE INCEPUT⟩ + ⟨LUNGIME⟩'
    I(+1) 'IN AFARA INTERVALULUI ?' ECRIRE SORTIE
FIN
```

```
M Y24 = :5:
SI Y24 = U
   ALORS
      I(+0) '03. VALOARE NUMERICA ?' ECRIRE SORTIE
FIN
EXEC LEXINSRT
M Y24 = :6:
SI Y24 = U
   ALORS
      I(+0) '03.VALOARE NENUMERICA ?' ECRIRE SORTIE
FIN
EXEC LEXINSRT
M Y24 = Y24 + :5:
SI Y24 < = 0 OU Y24 > 31
   ALORS
      I(+0) '04. <POZITIE INCEPUT> + <LUNGIME>'
      I(+1) 'IN AFARA INTERVALULUI ?' ECRIRE
      SORTIE
FIN
EXEC INSRT /* INSERAREA PROPRIU ZISA */
   M Z7 = Z7 (1  1)
   SI Z7 = '?'
      ALORS
         I(1) '**INSRT**ATT. VALOAREA TRANSMISA'
         I(+1) 'DE <EMITOR> A FOST TRUNCHIATA' I (+1) 'LA DREAPTA ?'
         ECRIRE
   FIN
FIN
:FDEF ?
```

— programul ⟨LERXINSRT⟩:

```
ZONCOM     DSECT
Y24        ORG      X'13C'
Z7         ORG      X'144'+7 * 32 + 1
LEXINSRT   CSECT
           USD,ZONCOM     11
           LD1,1          Z7
           SRL4,1         24
           LD1,0          Y24+3
           ST1,0          Z7,1
           AD4I,1         1
           SLL4,1         24
           ST1,1          Z7
           ST1,1          Z7 − 1
           BRU            *8*4
           END            LEXINSRT
```

Acest program construiește în Z_7 o structură de forma:

p	Z_r	p_1	l_1	Z_e	p_2	l_2

- l_2 → lungime emițător;
- p_2 → poziție început emițător;
- Z_e → număr variabilă emitoare;
- l_1 → lungime zona receptoare;
- p_1 → poziția de început în receptor;
- Z_r → număr variabilă receptoare;
- p → pointer general pe zonă și contor număr caractere.

Toate valorile sînt stocate în format binar pe un octet.

– programul ⟨INSRT⟩ utilizează valorile poziţionate de ⟨LEXINSRT⟩ şi realizează efectiv inserarea

Exemplu:

```
ZONCOM   DSECT       X'144'
ZI       ORG
Z7       ORG         X'144' + 7 * 32 + 1
INSRT    CSECT
         USD,ZONCOM  11
         BRU         BEGIN
MASCL    RES,4       0
         DATA,4,4    X'JA0AF040'
BEGIN    RES,4       0
         LD1,0       Z7 + 4      /* ⟨emitor⟩
         SRL4,0      24          /* cadrare dreapta
         SLL4,0      5           /* aflare număr ZI
         LD1,1       Z7+5        /* ⟨poziţie început₂⟩
         SRL4,1      24          /* cadrare dreapta
         AD4,0       1 *4        /* de unde începe transferul
         LD1,1       Z7 + 6      /* ⟨lungime₂⟩
         SRL4,1      24          /* lungime de transferat
         LD1,2       Z7+1        /* ⟨receptor⟩
         SRL4,2      24          /* cadrare dreapta
         SLL4,2      5           /* aflare număr Zi
         LD1,3       Z7+2        /* ⟨poziţie început₁⟩
         SRL4,3      24          /* cadrare dreapta
         AD4,2       3*4         /* poziţia de inserare efectivă
         LD1,3       Z7+3        /* ⟨lungime₁⟩
         SRL4,3      24          /* cadrare dreapta
         CP4,1       3*4         /* compară ⟨lungime₁⟩ cu ⟨lungime₂⟩
         BLEZ        MOVE        /* ⟨lungime₁⟩ mai mică sau egală cu
                                    ⟨lungime₂⟩
                                    dacă
         LD4,1       3*4         /* este mai mare atunci o reduc şi
         LD1I,4      '?'         /* marchez eroarea de atenţionare
         ST1,4       Z7          /* în primul octet al lui Z7
MOVE     RES,4       0
         SB4,3       1*4         /* eventual rest de umplut cu spaţii
         MVSR,0      *2*4        /* mută din ⟨emitor⟩ în ⟨receptor⟩
         LD4,0       1*4         /* adresa unde a ajuns în ⟨receptor⟩
         LD4,1       3*4         /* rest de tratat
         CYBR,0      X'40'       /* pune spaţii în acest rest
         LD1,0       Z7+1        /* receptor
         SRL4,0      24          /* cadrare dreapta
         SLL4,0      5           /* aflare număr Zi
         LD4,2       0           /* conservă adresa de început
         SB4I,2      1           /*
         LD4I,1      30          /* lungime şir de analizat
         AD4,0       1*4         /* analiza de la stînga la dreapta? —
         TRTL,0      MASCL       /* determină poziţia ultimului caracter
                                    introdus
         SLL4,1      24          /* şi o marchează în
         ST1,1       *2*4        /* receptor
         BRU         *8*4        /* întoarcere la apelant
         END         INSRT
```

Exemple de utilizare:

M Z1 = 'FUNCTIA LEXINSRT PERMITE' M Z2 = 'INSERT:'
INSERT Z1 (9 8) = Z2 (1 6)

I Z1 ?
Z1 : FUNCTIA ⌊_⌋ INSERT ⌊_⌊_⌊_⌊_⌋ PERMITE
INSERT Z1 (16 9) = Z1 (18 7) I Z1 ?
Z1 : FUNCTIA ⌊_⌋ INSERT ⌊_⌋ PERMITE

Modificări ale macrogeneratorului în versiune V 1.6.R.

Macrogeneratorul permite definirea unui formal temporar la execuţie prin introducerea blocului : DEFORM.

Sintaxa :

: DEFPRO ⟨*nume program*⟩ [: DEFPW ⟨*nume program control*⟩]
[: DEFROM
DEBUT
⟨*definire formal logic*⟩
FIN]
: CONTXT [⟨*citaţie formală* X_1⟩]
: EXP
⟨*model expansiune*⟩
: FDEF ?

Specificaţii de utilizare :

— formalul temporar trebuie să fie definit identic în toate subrutinele utilizate de program ;

— formalele temporare trebuie să fie specificate în aceeaşi ordine în toate subrutinele ;

— în blocurile : DEFORM şi : CONTXT este admisă utilizarea macroinstrucţiunilor.

Sintaxa de apel a programelor precompilate este :

EXEC {ENCOURS/⟨*nume program*⟩}
[(⟨*parametrii intrare*⟩ ; ⟨*parametrii ieşire*⟩)]

⟨*parametrii intrare*⟩ : : = $\langle V_i\rangle[\langle N_i\rangle]/\langle V_i\rangle[N_i]$ ⟨*parametrii intrare*⟩

⟨*parametrii ieşire*⟩ : : = $\langle V_i\rangle[\langle N_j\rangle]/\langle V_i\rangle[\langle N_j\rangle]$ ⟨*parametrii ieşire*⟩

$\langle V_i\rangle$: desemnează una din variabilele (în această ordine) X_1, Y_1, Z_1 sau W_1 ;

$\langle N_j\rangle$: specifică numărul variabilei de acelaşi tip care va primi valoarea parametrului ;

$\langle V_i\rangle$ din apelant se transmite în V_iN_i din subprogram.

— EXEC ENCOURS cheamă recursiv apelantul ;

La această instrucţiune variabilele X_1 trebuie să conţină la sfîrşitul programului valorile cu care au intrat în el (din punct de vedere logic). Celelalte variabile X_1, Z_1, W_1 sînt considerate variabile globale şi orice modificare asupra conţinutului este transmisă la sfîrşitul programului apelant. În cazul utilizării unor variabile locale în cadrul subprogramului este necesară salvarea valorilor transmise de apelant şi refacerea lor la retur.

Pentru a realiza acest lucru se va indica modul de transmitere a variabilelor subprogramului şi modul de întoarcere a lor. În sintaxa de apel variabilele definite între '(' şi ';' sînt variabile transmise de apelant iar cele cuprinse între ';' şi ')' variabile de retur.

Specificaţii de utilizare a parametrilor :

— la apel trebuie respectată succesiunea variabilelor de acelaşi tip ;

Exemplu :

EXEC PROG (X1 2 X3 5 Y3 1 ; Z1 4 Z2 1 W2 5) are ca efect : (X1 → X2, X3 → X5, Y3 → Y1 ; Z1 → Z4, Z2 → Z1, W2 → W5) ;

— dacă una sau mai multe variabile de tip X_i conțin(e) adresa unui (unor) buffer(e) ordinar(e) sau de sortare atunci aceste variabile vor fi transmise în mod obligatoriu subprogramului și vor fi returnate programului apelant ;

— în cazul succesiunii unor variabile ⟨V_i⟩ de același tip variabilele corespondente vor fi specificate în ordinea crescătoare a lui ⟨N_j⟩ ;

— la absența unei valori ⟨N_j⟩ valoarea implicită a acestuia este prima valoare neutilizată ;

Exemplu :

EXEC PROG (X1 2 X5 X6 ; X2) are ca efect :
(X1 → X2, X5 → X1, X 6 → X3 ; X2 → X1)
Schimbul de parametrii pentru apelul :
EXEC PROG (X2 X1 3 Y7 7 Y8 Y1 8 Z2 1 W4 ;
X4 Y3 Y1 8 W6)
poate fi ilustrat astfel :

⟨apel⟩			⟨retur⟩	
⟨apelant⟩	*⟨subprogram⟩*	|	*⟨apelant⟩*	*⟨subprogram⟩*
X2 →	X1	|	X4 ←	X1
X1 →	X3	|	Y3 ←	Y1
Y7 →	Y7	|	Y1 ←	Y8
Y8 →	Y1	|	W6 ←	W1
Y1 →	Y8	|		
Z2 →	Z1	|		
W4 →	W1	|		

Pentru construirea programelor IMT zona de comunicație a fost modificată, pentru a permite accesul la variabilele de lucru W_i, astfel :

```
ZONCOM   DSECT
Yi       ORG     X'OEO,
Y1       RES,4   1
Y2       RES,4   1
...      ...     ...
Y25      RES,4   1
PAROLA   ORG     X'144'
PW       RES,4   1
Zi       ORG     X'164'
Z1       RES,4   8
Z2       RES,4   8
...      ...     ...
Z7       RES,4   8
Wi       ORG     X'498'
W1       RES,4   3
W2       RES,4   3
...      ...     ...
W14      RES,4   3
```

Programele IMT construite sînt catalogate (definite) în spaţiul „PROG" al bazei de date utilizînd sintaxa :

: DEFIMT ⟨*nume program*⟩ [⟨*parametrii bibliotecii*⟩] ?

[⟨*cartele* IMT⟩]

Dacă ⟨*parametrii bibliotecii*⟩ nu sînt specificaţi atunci programul IMT este sub formă de ⟨*cartele* IMT⟩.

⟨*parametrii bibliotecii*⟩ : sînt parametrii de identificare a bibliotecii şi codului IMT al programului. Între parametrii se utilizează ca separator spaţiu. Structura acestui element este :

DV : AD** LN : ⟨*nume bibliotecă*⟩ FN : ⟨*nume program*⟩ [*GN* : *g* UN : *v* UN : *u*]

** — numărul logic al discului pe care se află biblioteca ;

⟨*nume bibliotecă*⟩ — numele bibliotecii ;

GN : *g* — numărul său de generare (implicit 0) ;

VN : *v* — numărul său de versiune (implicit 1) ;

⟨*nume program*⟩ : numele programului IMT ;

UN : *u* — varianta programului (implicit UN : 255 — cea mai recentă versiune).

Editorul de texte

Editorul de texte SOCRATE asigură gestiunea textelor programelor SOCRATE şi a caracteristicilor de tip ⟨*text*⟩ definite în structura bazei de date.

Funcţiunile editorului de texte permit :

— crearea, listarea, punerea la zi a textelor ;

— inserarea de noi linii ;

— ştergerea unor linii ;

— substituirea unor şiruri de caractere la nivel de linie sau la nivelul întregului text ;

— gestionarea fişierelor de test ale definirii structurilor şi programelor cu ajutorul unui bibliotecar ;

— accesul la textele sursă ale macro sau programelor precompilate.

Ediţia unui fişier structură, program de test, macroinstrucţiune sau program compilat trebuie să fie precizată explicit, după apelul editorului, printr-o comandă de căutare a fişierului.

Ediţia caracteristicilor de tip ⟨*text*⟩ din baza de date trebuie să fie precizată printr-o cerere de punere la zi (M ⟨*text*⟩ ⟨*calificare*⟩ = EXT) pentru care compilatorul de cereri (R, O) generează un apel la editorul de texte care devine efectiv la momentul execuţiei.

La introducerea unui fişier structură, a unei macroinstrucţiuni a unui program de cereri sau a unui text apelul editorului se realizează prin tastarea caracterului EXT (sau în coloana 1 în „batch processing"). În conversaţional editorul de texte anunţă aşteptarea unei comenzi prin „promptul", " —", în coloana 1 a liniei curente, urmat de cursor.

Comenzile editorului de texte se împart în două categorii :

— comenzi la nivel de fişier ;

— comenzi la nivel de linie.

Orice text (program de cereri, macroinstrucţiune, program precompilat sau fişier structură) care precede apelul editorului de texte se numeşte fişier curent.

Fişierul curent este accesibil funcţiunilor editorului de texte după apelul acestuia. Acestui text curent i se poate substitui orice fişier, al bibliotecarului editorului de texte sau al macrogeneratorului, prin apelarea explicită.

După execuţia unei comenzi la nivel fişier, editorul de texte este pus în aşteptarea unei comenzi la nivel de linie. Dacă în cursul aşteptării unei comenzi la nivel de linie se detectează o comandă la nivel de fişier, atunci va fi executată această comandă (încărcînd fişierul curent citat) şi se trece la aşteptarea unei comenzi de linie.

Editorul de texte este astfel construit încît este admisă interacţiunea sa cu celelalte componente astfel:
- interacţiunea „Editor-Editor" (fig. 5.1);
- interacţiunea „Editor-compilator LDD" (fig. 5.2);

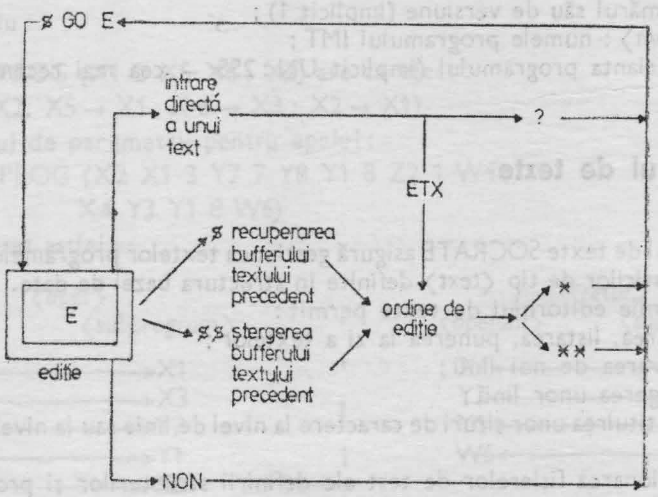


Fig. 5.1. Interacţiunea „Editor-Editor"

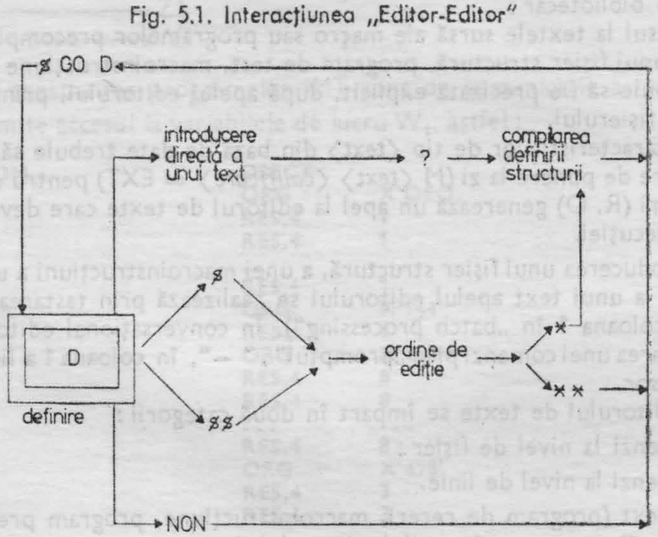


Fig. 5.2. Interacţiunea „Editor-Compilator LDD"

— interacţiunea „Editor-procesor creare" (fig. 5.3) ;
— interacţiunea „Editor-compilator LMD" (fig. 5.4) ;
— interacţiunea „Editor-Macrogenerator" (fig. 5.5).

Aceste interacţiuni fac din editor un suport pentru programare interactivă. În conversaţional un procesor lansat îşi anunţă prezenţa prin trimiterea promptului QUESTION :

Dacă răspunsul la acest prompt este unul din caracterele specificate în figurile prezentate atunci va fi lansată acţiunea asociată acestuia. Dacă utilizatorul introduce direct de la terminal un text destinat unui procesor atunci posibilitatea introducerii unei noi linii este marcată prin promptul „*".

După apelul unui procesor (definire, cereri sau macrogenerator) utilizatorul are posibilitatea introducerii directe a textului corespunzător prin perifericul de intrare. Sfîrşitul acestui text sursă este semnalat prin prezenţa caracterului „?" ca ultim caracter semnificativ diferit de spaţiu al unei linii.

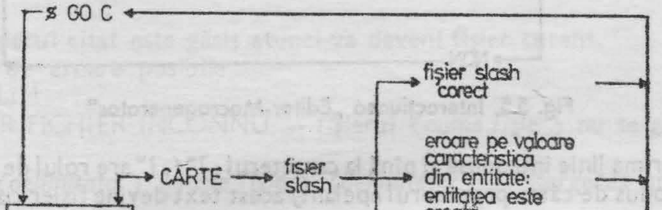


Fig. 5.3. Interacţiunea „Editor-procesor creare"

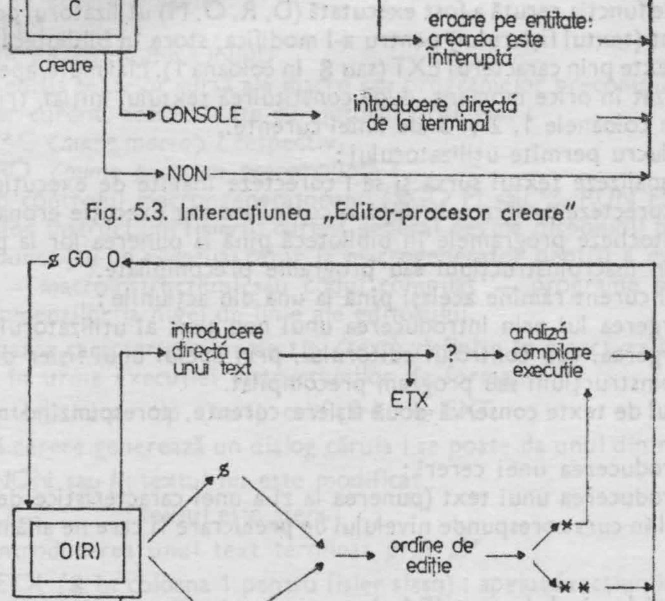


Fig. 5.4. Interacţiunea „Editor-compilator LMD"

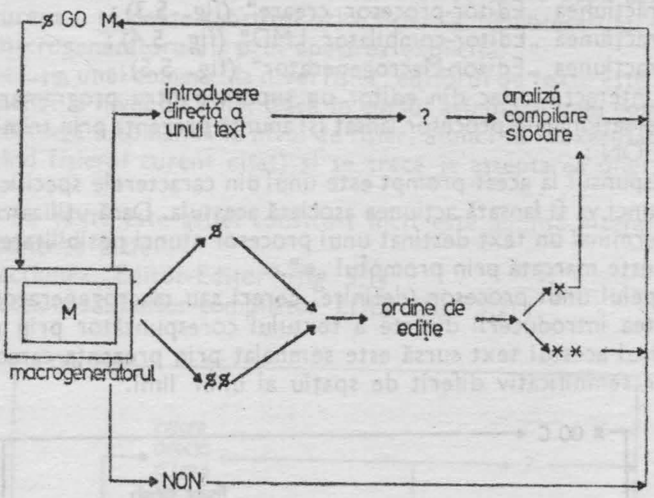


Fig. 5.5. Interacţiunea „Editor-Macrogenerator"

De la prima linie introdusă şi pînă la caracterul „?" („?" are rolul de a lansa analiza textului introdus de către procesorul apelant) acest text devine fişier curent ştergînd fişierul curent precedent.

După ce funcţia cerută a fost executată (D, R, O, M) utilizatorul poate recupera fişierul curent (textul fişierului) pentru a-l modifica, stoca în bibliotecă etc., apelînd editorul de texte prin caracterul EXT (sau $ în coloana 1). Mai mult, apelul editorului poate fi realizat în orice moment, după constituirea textului iniţial, trimiţînd caracterul EXT în coloanele 1, 2 şi 3 ale liniei curente.

Acest lucru permite utilizatorului :

– să analizeze textul sursă şi să-l corecteze înainte de execuţie ;

– să corecteze programele după o compilare sau execuţie eronată ;

– să stocheze programele în bibliotecă pînă la punerea lor la punct şi să le transforme în macroinstrucţiuni sau programe precompilate.

Fişierul curent rămîne acelaşi pînă la una din acţiunile :

– ştergerea lui prin introducerea unui nou text al utilizatorului ;

– ştergerea, sub controlul editorului, prin apelul unui fişier din bibliotecă, a unei macroinstrucţiuni sau program precompilat.

Editorul de texte conservă două fişiere curente, corespunzînd nivelurilor de prelucrare :

– introducerea unei cereri ;

– introducerea unui text (punerea la zi a unei caracteristice de tip ⟨text⟩).

Fişierul în curs corespunde nivelului de prelucrare la care ne aflăm.

Comenzi la nivel de text (fişier)

Utilizatorul are posibilitatea să stocheze textele sursă (definiri de structuri, programe de testare etc.) cu ajutorul bibliotecarului editorului de texte. Comenzile la nivel fişier sînt următoarele :

1) •L — lista fişierelor stocate în bibliotecă ;

2) *R ⟨*delimitator*⟩ ⟨*nume fişier*⟩ ⟨*delimitator*⟩

— ⟨*delimitator*⟩ : un caracter alfanumeric ;

— ⟨*nume fişier*⟩ : maxim 11 caractere alfanumerice ;

— R : catalogarea unui fişier în bibliotecă.

Ultimul caracter din ⟨nume fişier⟩ trebuie să difere de delimitator pentru a nu fi confundat cu acestea.

Exemplu :

*R/FISEDIT/

*R ⟨EXEMPLU⟩

Mesaje de eroare :

ER DELIM : cel de-al doilea delimitator n-a fost găsit ;

FICHIER DEJA CREE VOULEZ.VOUS L'ECRASER? : fişierul există, doriţi să-l distrugeţi (OUI — da ; NON — nu)

3) *I ⟨*delimitator*⟩ ⟨*nume fişier*⟩ ⟨*delimitator*⟩ — apelul unui fişier prezent în bibliotecă.

Dacă fişierul citat este găsit atunci va deveni fişier curent.

Mesaje de eroare posibile :

ER DELIM

ERREUR FICHIER INCONNU — fişierul ⟨*nume fişier*⟩ nu se găseşte în bibliotecă.

4) *D ⟨*delimitator*⟩ ⟨*nume fişier*⟩ ⟨*delimitator*⟩ — ştergerea unui fişier din bibliotecă.

Mesaje de eroare posibile :

ER DELIM

ERREUR FICHIER INCONNU

Pentru ediţia textelor sursă ale macro sau programelor precompilate se transformă în fişier curent, aceste texte cu ajutorul ordinelor :

: EDIMAC ⟨*nume macro*⟩ ? respectiv,

: EDIPRO ⟨*nume program precompilat*⟩ ?

executate sub controlul macrongeneratorului ($GO M sau % RUN FN : M). După execuţia acestor instrucţiuni fişierul curent general devine disponibil pentru listare şi punere la punct (cu un eventual retur la macrogenerator pentru a modifica codul intermediar — macroinstrucţiuni sau codul compilat — programe precompilate) cu ajutorul comenzilor la nivel de linie ale editorului.

Actualizarea caracteristicilor de tip ⟨text⟩ definite în structura bazei de date se realizează în urma execuţiei instrucţiunilor de forma :

M caracteristici de tip ⟨text⟩ calificare ; = EXT.

Această cerere generează un dialog căruia i se poate da unul din răspunsurile :

a) — NON sau / : textul nu este modificat ;

b) — U : textul este şters ;

c) — introducerea unui text terminat prin „?" ;

d) — ETX ($ în coloana 1 pentru fişier slash) : apelul funcţiunilor editorului de texte pentru a corecta textul prezent în baza de date.

Deoarece c) şi d) sînt recursive ele se vor termina prin ordinul *END*.

Această instrucţiune scoate în evidenţă necesitatea existenţei a două fişiere curente, şi anume :

— unul afectat procesoarelor ;

— celălalt, afectat editorului de texte. Necesitatea derivă din aceea că programele de testare, macroinstrucţiunile şi programele precompilate pot face apel la editor prin cereri de forma prezentată.

Comenzi la nivel de linie

Comenzile la nivel de linie, ale editorului de texte au următoarea formă sintactică generală :

[⟨*început*⟩] [,⟨*sfîrşit*⟩] ⟨*comandă*⟩ [⟨*specificator*⟩]
[⟨*delimitator*⟩ ⟨*criteriu*⟩ ⟨*delimitator*⟩]
[⟨*şir de caractere*⟩]

Liniile textului sînt reperate prin :

— numărul lor de ordine ;

— conţinutul lor cu ajutorul unui criteriu de selecţie.

Majoritatea comenzilor acţionează asupra unui grup de linii delimitate prin ⟨*început*⟩ şi ⟨*sfîrşit*⟩ ; şi mai mult, din acest gurp de linii pot fi selectate numai acelea care îndeplinesc un ⟨*criteriu*⟩.

Numărul liniei [⟨*început*⟩/⟨*sfîrşit*⟩] exprimă poziţia sa în raport cu începutul textului (prima linie are numărul 1). Mai mult, inserarea sau ştergerea unor linii antrenează renumerotarea celor care le urmează în text.

⟨*început*⟩ : numărul liniei de la care începe să se desfăşoare acţiunea definită de ⟨*comandă*⟩ ;

⟨*sfîrşit*⟩ : numărul liniei pînă la care (inclusiv) se desfăşoară acţiunea definită de ⟨*comandă*⟩ ;

⟨*criteriu*⟩ : şir de caractere, cuprins între două caractere identice (⟨*delimitator*⟩), care permit desemnarea unor/unei linii prin conţinut ;

⟨*delimitator*⟩ : un caracter alfanumeric care diferă de orice caracter din şirul care defineşte ⟨*criteriu*⟩ ;

După fiecare operaţie editorul punctează ultima linie tratată care devine astfel linie curentă (dacă ⟨*început*⟩ şi ⟨*sfîrşit*⟩ nu sînt specificate acţiunea comenzii editorului se desfăşoară asupra liniei curente).

⟨*început*⟩ : : = număr absolut/±număr 0 (zero)

— număr absolut : numărul liniei dorite ;

— ± număr : distanţa liniei dorite faţă de linia curentă. Noua linie curentă este definită de : număr linie curentă ± număr ;

— 0 : marchează începutul fişierului curent.

⟨*sfîrşit*⟩ : : = număr absolut/+ număr/$

— **$** : marchează ultima linie definită în fişierul curent.

Vom prezenta în continuare sintaxa comenzilor la nivel de linie :

1) **Comanda de listare cu număr** :

— [⟨*început*⟩] [,⟨*sfîrşit*⟩] L [[A] ⟨*delimitator*⟩ ⟨*criteriu*⟩ ⟨*delimitator*⟩]

Permite listarea, conform criteriului definit, a liniilor cuprinse între ⟨*început*⟩ şi ⟨*sfîrşit*⟩. Liniile sînt scrise pe suportul de ieşire şi precedate de o linie care conţine numărul lor. În cazul în care unele din elementele opţionale lipsesc, următoarele variante sintactice ne dau :

— L : lista tuturor liniilor fişierului ;

– L ⟨*delimitator*⟩ ⟨*criteriu*⟩ ⟨*delimitator*⟩ : lista primei linii care conţine ⟨*criteriu*⟩ ;

– LA ⟨*delimitator*⟩ ⟨*criteriu*⟩ ⟨*delimitator*⟩ : lista tuturor liniilor care conţin ⟨*criteriu*⟩.

Exemplu :

– 1,$ LA*ASE* : caută şi listează toate liniile care conţin cuvîntul ASE.
Mesaje :
EOF HIT : a fost atins sfîrşitul fişierului cerut ;
NONE : nu a fost găsită nici-o linie care să satisfacă ⟨*criteriu*⟩ sau poziţia definită de ⟨*început*⟩, eventual de ⟨*sfîrşit*⟩).
?**? : nu a fost recunoscut ordinul (eroare de sintaxă).

Aceste mesaje pot să apară la oricare din comenzile la nivel de linie.

2) **Comanda–de listare fără număr :**

– [⟨*început*⟩] [,⟨*sfîrşit*⟩] P [[A]⟨*delimitator*⟩ ⟨*criteriu*⟩ ⟨*delimitator*⟩]

Modul de acţiune este similar cu cel al comenzii L numai că linia care conţine numărul nu se imprimă.

3) **Comanda de căutare a numerelor :**

– ⟨*început*⟩] [,⟨*sfîrşit*⟩] N [[A]⟨*delimitator*⟩ ⟨*criteriu*⟩ ⟨*delimitator*⟩]

Se imprimă numai linia care conţine numărul.

4) **Comanda de substituire**

– [⟨*început*⟩] [,⟨*sfîrşit*⟩] S[[A] ⟨*delimitator*⟩ ⟨*criteriu*⟩ ⟨*delimitator*⟩ ⟨*şir caractere*⟩ ⟨*delimitator*⟩]

Permite substituirea lui ⟨*criteriu*⟩ cu ⟨*şir caractere*⟩ în toate liniile cuprinse între ⟨*început*⟩ şi ⟨*sfîrşit*⟩.

– S ⟨*delimitator*⟩ ⟨*criteriu*⟩ ⟨*delimitator*⟩ ⟨*şir caractere*⟩ ⟨*delimitator*⟩ : substituirea se realizează asupra primului criteriu detectat ;

– SA ⟨*delimitator*⟩ ⟨*criteriu*⟩ ⟨*delimitator*⟩ ⟨*şir caractere*⟩ ⟨*delimitator*⟩ : substituţia se realizează asupra tuturor ⟨*criteriu*⟩ întîlnite în textul analizat.

După fiecare substituire linia modificată este imprimată împreună cu numărul său. Dacă o substituire antrenează depăşirea liniei atunci se va efectua substituţia şi se va imprima conţinutul liniei înainte şi după substituire.

5) **Comanda de ştergere :**

– [⟨*început*⟩] [,⟨*sfîrşit*⟩] D

Permite ştergerea liniilor cuprinse între ⟨*început*⟩ şi ⟨*sfîrşit*⟩ (inclusiv acestea). După efectuarea comenzii linia curentă este indicată de ⟨*început*⟩.

6) **Comanda de inserare.**

– [⟨*început*⟩] I

Permite inserarea (după linia desemnată de ⟨*început*⟩) între două linii consecutive a fişierului curent a uneia sau mai multor linii introduse prin perifericul de intrare. Inserarea se întrerupe cu ajutorul caracterului ETX (sau $ în coloana 1). Pointerul curent al liniilor este mărit cu un număr egal cu acela al liniilor inserate.

7) **Comanda de inserare a unei singure linii :**

– [⟨*început*⟩] IC ⟨*şir de caractere*⟩

Permite inserarea, după linia punctată de ⟨*început*⟩ a liniei definită prin ⟨*şir caractere*⟩.

8) **Comanda de înlocuire**

– [⟨*început*⟩] [,⟨*sfîrşit*⟩] R

Permite înlocuirea liniilor cuprinse între ⟨*început*⟩ şi ⟨*sfîrşit*⟩ cu linii introduse de la perifericul de intrare.

Comanda este echivalentă cu secvenţa :

– [⟨*început*⟩] [,⟨*sfîrşit*⟩] D
– 1I
•
•
• linii inserate
•
•
– END

Următoarea secvenţă echivalează cu o ştergere :

– [⟨*început*⟩] [,⟨*sfîrşit*⟩] R
– ETX

9) Comanda de înlocuire a unei linii :

– [⟨*început*⟩] D
– 1IC ⟨*şir de caractere*⟩

Dacă ⟨*început*⟩ = ⟨*sfîrşit*⟩ atunci comanda se desfăşoară numai asupra liniei indicate.

10) Comanda de oprire a editorului :

– END

Comanda opreşte ediţia şi trece controlul procesorului următor.

Exemple :

```
$ GO E
EDITION:
* ACESTA ESTE UN EXEMPLU
* DE UTILIZARE A EDI
* AL SGBD SOCRATE
* ?
$REST
EDITION:$
— 1, $L
   1
   2
ACESTA ESTE UN EXEMPLU
   3
DE UTILIZARE A EDI
   4
AL SGBD SOCRATE
   5
?
EOF HIT
— 1R ********
* ETX
— 1, $L
   1
ACESTA ESTE UN EXEMPLU
   2
DE UTILIZARE A EDI
   3
AL SGBD SOCRATE
   4
EOF HIT
```

```
— 2SA /EDI/EDITORULUI DE TEXTE/
  2
DE UTILIZARE A EDITORULUI DE TEXTE
— 4I
* FĂRĂ INTERACTIUNE CU CELELALTE COMPONENTE
* ETX
— 1, $P
ACESTA ESTE UN EXEMPLU
DE UTILIZARE A EDITORULUI DE TEXTE
AL SGBD SOCRATE
?
FARA INTERACTIUNE CU CELELALTE COMPONENTE
EOF HIT
— 4L
  4
?
— 4D
— 4L
  /:
FĂRĂ INTERACTIUNE CU CELELALTE COMPONENTE
EOF HIT
— *R /EXEMPLU /CATALOGARE TEXT EDI
— 1,$D STERGE FISIER CURENT
— 1,$L (S-A STERS ?)
ERDELIM
— 1,$L
EOF HIT
—*I /EXEMPLU/RECUPERARE TEXT
— 1,$P
ACESTA ESTE UN EXEMPLU
DE UTILIZARE A EDITORULUI DE TEXTE
AL SGBD SOCRATE
FARA INTERACTIUNE CU CELELALTE COMPONENTE
EOF HIT
—
LIGNVID
— *1
* ??? *
— *L
ABC
EXEMPLU
EOF TXT
— *I /ABC/
— 1,$P
*
TTTT
RRRR
?
EOF HIT
— *I /EXEMPLU/
— $1
* ale sistemului deoarece este un TEXT oarecare
*
* ETX
— $LA
  6
etx
EOF HIT
— — 1P
ale sistemului deoarece este un TEXT oarecare
— 5, —1P
ABS DEC
```

```
— —2,3P
AL SGBD SOCRATE
— 3P
AL SGBD SOCRATE
— —2P
ACESTA ESTE UN EXEMPLU
— —1D
— —1L
— 1P
DE UTILIZARE A EDITORULUI DE TEXTE
— —1R
* acesta este un scurt exemplu
* ETX
— —1P
— 1,$P
acesta este un scurt exemplu
AL SGBD SOCRATE
FARA INTERACTIUNE CU CELELALTE COMPONENTE
ale sistemului deoarece este un TEXT oarecare
etx
EOF HIT
— 1,1SA *exemplu*exemplu de utilizare al EDITORULUI *
ERDELIM
— 1SA (exemplu 〈EXEMPLU DE UTILIZARE〉
  1
acesta este un scurt EXEMPLU DE UTILIZARE
— 2SA *AL* a funcţiunilor EDITORULUI DE TEXTE AL*
  2
a funcţiunilor EDITORULUI DE TEXTE ALE SGBD SOCRATE
— 1,$P
acesta este un scurt EXEMPLU DE UTILIZARE
  a funcţiunilor EDITORULUI DE TEXTE AL SGBD SOCRATE
FARA INTERACTIUNE CU CELELALTE COMPONENTE
ale sistemului deoarece este un TEXT oarecare
etx
EOF HIT
— $D
— *R /EXEMPLU/
ERREUR FICHIER DEJA CREE
— *D /EXEMPLU/
— *R /exemplu/
— *L
ABC
exemplu
EOF TXT
— *D /ABC/
— *I exemplu
— *1
* ??? *
— *1
exemplu
EOF TXT
— *R /EXEMPLU/
— *L
EXEMPLU
exemplu
EOF TXT
— *I EXEMPLU
— 1,$PA
acesta este un scurt EXEMPLU DE UTILIZARE
  a funcţiunilor EDITORULUI DE TEXTE AL SGBD SOCRATE
```

```
FARA INTERACTIUNE CU CELELALTE COMPONENTE
ale sistemului deoarece este un TEXT oarecare
EOF HIT
— &
$GO R
QUESTION:
* M Y1 = EXT M Y2 = EXT M Y1 = Y1 + Y2/Y1 I Y1 I Y2 ?
Y1 :321
Y2 :213
Y1 :321
Y2 :213
QUESTION: $
— 1,$L
   1
   2
M Y1 = EXT M Y2 = EXT M Y1 = Y1 + Y2/Y1 I Y1 I Y? ?
EOF HIT
— *
Y1 : 2
Y2 : 2
Y1 : 3
Y2 : 2
QUESTION: $
— *
Y1 : 44
Y2 : 44
Y1 : 45
Y2 : 44
QUESTION : NON
$GO E
EDITION:$
— *L
EXEMPLU
exemplu
EOF TXT
— *D /exemplu/
— *1
* ??? *
— *L
EXEMPLU
EOI TXT
— 1, $PA
M Y1 = EXT M Y2 = EXT M Y1 = Y1 + Y2/Y1 I Y1 Y2 ?
EOF HIT
— *I [EXEMPLU]
— 1,$PA
acesta este un scurt EXEMPLU DE UTILIZARE
a funcţiunilor EDITORULUI DE TEXTE ALE SGBD SOCRATE
FARA INTERACTIUNE CU CELELALTE COMPONENTE
ale sistemului deoarece este un TEXT oarecare
EOF HIT
$GO
QUESTION:
* I Y1  I Y2 I Y25 ?
Y1 : 45
Y2 : 44
Y25 : 0
QUESTION: $
— *R /PGM1/
— *
Y1 : 45
Y2 : 44
```

```
Y25 : 0
QUESTION: $
— 1P
— 2P
I Y1 I Y2 I Y25 ?
EOF HIT
— *
Y1 : 45
Y2 : 44
Y25 : 0
QUESTION:$
— 1,$D
— *
*ER,R21 NL 1 ...? ... AU LIEU DE CODF. D ACTION
ERREUE EN REQUETE
QUESTION:NON
$GO R
QUESTION:$
— *I /PGM1/
— *
Y1 : 45
Y2 : 44
Y25 : 0
QUESTION:$
— ?
* ??? *
— $$
* ??? *
— *D (PGM1(
— *L
EXEMPLU
EOF TXT
— *I 〈EXEMPLU〉
— *
*ER.R21 NL 1 ... acesta ... AU LIEU DE CODE D ACTION
ERREUR EN REQUETE
QUESTION:NON
$CLOCK
18.49.25.62.
$MAIL
RIEN A SIGNALER
$NUMBER
   1
$MESSOP SESIUNE NORMALA ?
$GO
QUESTION:$
— *L
EXEMPLU
EOF TXT
— *D /EXEMPLU/
— *L
EOF TXT
```

Interfaţa cu limbajele evoluate

Reprezintă o metodă de acces la datele conţinute de baza de date a cărei introducere a fost cerută, în principal de :

— durata mare a unor prelucrări ;

— volumul de informaţii la intrare sau ieşire, provenienţa sau destinaţia informaţiilor etc.

Metoda de acces este pusă la dispoziţia programelor scrise în diferite limbaje (COBOL, FORTRAN, ASSIRIS etc.) şi constă în aceea că fiecare cerere de acces, a unui program, la baza de date se realizează prin intermediul unui modul de legătură standard.

Metoda definită respectă principiul independenţei programelor faţă de date prin :

— definirea unor primitive de acces, analoge celor de gestiune a fişierelor clasice, care permit accesul la toate caracteristicile elementare ale bazei conform oricărei căi de parcurgere definită în structură ;

— simplitatea utilizării în limbaje evoluate ;

— modificările structurii bazei de date nu necesită recompilarea programelor batch deoarece modulul dă controlul unor subprograme scrise în limbaj SOCRATE, (programe precompilate) care realizează funcţiunile dorite (se modifică, eventual, numai aceste subprograme).

Vom prezenta în continuare punctele de intrare în modulul de legătură (subprograme apelabile din limbajul evoluat utilizat conform regulilor de apel specifice acestora). Pentru sintaxa de apel a subprogramelor şi modul de descriere al argumentelor vom face referire la limbajele COBOL, ASSIRIS şi FORTRAN.

Semnificaţia şi parametrii punctelor de intrare

1. Punctul de intrare SOPEN

SOPEN ⟨FISIER, TIP, NPC, NCP, NCD, NBS⟩

SOPEN : deschiderea unei baze de date ;

FISIER : identificatorul unei zone de memorie care defineşte complet baza de date la care se va efectua accesul.

Structura acestei zone este următoarea :

FISIER ⟨FIL1, STRU, UTI, AN, PW, ERR ⟨CODER, SUBRER⟩, FIL2⟩

— FIL1 : 4 caractere cu valoarea spaţiu ;
— STRU : 12 caractere α-n cu valoare ⟨*nume-bază*⟩ ;
— UTI : 8 caractere α-n cu valoare ⟨*nume-uti*⟩ ;
— AN : 4 caractere α-n cu valoare ⟨*cont-uti*⟩ ;
— PW : 8 caractere α-n cu valoare ⟨*parola*⟩ ;
— CODER : 1 cuvînt binar cu semn ;
— SUBRER : 30 caractere α-*n* ;
— FIL2 : 58 caractere α-*n* ;

⟨*nume-bază*⟩ : numele bazei de date la care se doreşte accesul (parametrul BN din comanda SOC) ;

⟨*nume-uti*⟩ : numele unui utilizator al acestei baze de date (parametrul PN din comanda SOC) ;

⟨*cont-uti*⟩ : numărul cont al utilizatorului (parametrul AN din comanda SOC) ;

⟨*parola*⟩ : parola utilizatorului (parametrul PW din cartela SOC).

Cîmpul ERR va conţine, după apel, informaţii privind execuţia subprogramului SOPEN astfel :

— **cîmpul** CODER (valoare în hexa) :

00 — nici o eroare ;

01 – sfîrşit de fişier ;
03 – realizarea de entitate deja prezentă (tentativă de creare sau generare a unei realizări prezente) ;
04 – entitate saturată (tentativa de creare sau generare a unei realizări după crearea sau generarea numărului maxim de realizări posibile) ;
07 – rang absolut în afara entităţii (tentativă de acces la o realizare căreia i se indică expres numărul iar acest număr depăşeşte numărul maxim de realizări posibile) ;
09 – realizare de entitate absentă ;
OE – cheie deja prezentă (tentativă de introducere, într-o caracteristică discriminantă, a unei valori introdusă anterior) ;
OF – cheie negăsită (valoarea atribuită cheii nu există în dicţionar) ;
52 – valoare numerică eronată (conţine caractere alfanumerice sau nu aparţine intervalului de valori declarat pentru caracteristică) ;
54 – valoare din lista eronată (valoarea nu este definită în lista de valori).

– **cîmpul SUBRER** : în cazul detectării unei erori cîmpul va conţine numele subprogramului SOCRATE care a provocat eroarea. Eventualele subprograme care urmează acestuia, în lista parametrilor de apel al unui punct de intrare, nu vor mai fi executate.

Prin analiza cîmpurilor CODER şi SUBRER programatorul poate să-şi construiască în programul său secvenţe de depanare a erorii semnalate.

• NBS : zonă de memorie de tip cuvînt binar iniţializată cu o ⟨valoare⟩ egală cu numărul blocului de structură prezent în memorie. Dacă ⟨valoare⟩ = 0 atunci sistemul atribuie automat lui NBS valoarea 1.

• TIP : zonă de memorie de 1 caracter a cărei valoare defineşte modul de deschidere al bazei de date.

Valorile atribuite zonei pot fi :
I – deschidere de interogare (citire) ;
O – deschidere în creare (scriere) ;
S – deschidere în interogare şi actualizare.

• NPC : zonă de memorie de tip cuvînt binar iniţializată cu o ⟨*valoare*⟩ care reprezintă numărul de puncte curente asociate bazei de date (numărul de variabile X_i iniţializate cu contextul FICHIER), unde ⟨*valoare*⟩ ∊ [0, 10]. Dacă ⟨*valoare*⟩ = 0 atunci sistemul atribuie lui NPC valoarea 10.

• NCP : zonă de memorie de tip cuvînt binar care conţine o ⟨*valoare*⟩ reprezentînd numărul de cadre program asociat bazei de date. Numărul de cadre program este reprezentat de numărul de pagini asociat celui mai mare program SOCRATE utilizat la apel, unde ⟨*valoare*⟩ ∊ [0,255]. Dacă ⟨*valoare*⟩ = 0 atunci sistemul atribuie automat lui NCP o valoare egală cu numărul de cadre necesar execuţiei celui mai mare subprogram din spaţiul PROG al bazei de date.

• NCD : zonă de memorie de tip cuvînt binar iniţializată cu o ⟨*valoare*⟩ reprezentînd numărul de cadre de date asociat bazei de date (parametrul CF al comenzii OPTION). Dacă ⟨*valoare*⟩ = 0 atunci sistemul atribuie automat lui NCD valoarea 10.

2) **Punctul de intrare SCLOSE**

SCLOSE ⟨FISIER⟩

SCLOSE : închiderea bazei de date desemnată de ⟨**FISIER**⟩ ;
FISIER : structura şi conţinutul sînt identice cu cele reprezentate la punctul 1.

3) **Punctul de intrare SSAVE**

SSAVE ⟨FIŞIER⟩

SSAVE : crearea unui punct de control, pe fişierul jurnal, pentru baza de date desemnată de ⟨FIŞIER⟩
FIŞIER : idem punctul 1.

4) Punctul de intrare STERM

STERM

STERM : terminarea sesiunii de lucru şi închiderea fişierului jurnal.

5) Punctul de intrare SGBD

SGBD ⟨FIŞIER, BUF, LGBUF, NRPGM, PGM1, PGM2,...,PGM_p, NRX,
NRX, PX1, PX2,...,PX_x,
NRY, PY1, PY2,...,PY_y
NRZ, PZ1, PZ2,...,PZ_z⟩

SGBD : acces la baza de date.

- FISIER : idem punctul 1 ;
- BUF : identificatorul unei zone de memorie de IGBUF octeţi, utilizată de subprogramele SOCRATE (definite de conţinutul lui PGM1,...,PGM_p) în operaţiile de intrare/ieşire.

Această zonă trebuie să fie structurată conform caracteristicilor FORMAL utilizate de subprogram.

Caracteristicile unei baze de date SOCRATE accesibile printr-un program batch, în interogare sau punere la zi, sînt de tip ⟨*cuvînt*⟩, ⟨*listă-de-valori*⟩ sau ⟨*valoare-numerică*⟩ (binar). Mărimea în octeţi a caracteristicilor este definită de descrierea de tip FORMAL cu care lucrează subprogramul iar corespondenţa caracteristicilor de tip SOCRATE cu tipurile de date admise de un limbaj evoluat este pentru :

- ⟨*cuvînt*⟩ şi ⟨*listă-de-valori*⟩ : date alfanumerice ;
- ⟨*valoare-numerică*⟩ : date reprezentate în zecimal împachetat (cu semn) sau zecimal despachetat.

Regulile de cadraj ale conţinutului caracteristicilor sînt :

- ⟨*cuvînt*⟩ şi ⟨*lista-de-valori*⟩ : se cadrează la stînga cu eventuale adăugări de spaţii EBCDIC ;
- ⟨*valoare-numerică*⟩ : cadrare la dreapta cu eventuale adăugări de zerouri nesemnificative :
  - binare pentru zecimal împachetat ;
  - EBCDIC pentru zecimal despachetat.

- IGBUF : zonă de memorie de tip cuvînt binar iniţializată cu o valoare egală cu numărul de caractere al zonei BUF. Dacă nu se utilizează zona BUF argumentului LGBUF i se dă valoarea 0 (zero).
- NRPGM : zonă de memorie de tip cuvînt binar iniţializată cu o ⟨*valoare*⟩ egală cu numărul ⟨*p*⟩ de subprograme apelate. Dacă valoarea atribuită lui NRPGM este 0 atunci argumentele PGM1, PGM2,...,PGM_p se omit. Dacă valoarea atribuită lui NRPGM este ⟨*p*⟩ atunci trebuiesc specificate ⟨*p*⟩ zone de memorie care conţin numele subprogramului apelat (PGM_i).
- PGM_i ($1 \leqslant i \leqslant p$) : identificatorul unei zone de memorie de tip alfanumeric cu lungimea de 30 caractere, iniţializată cu o valoare egală cu *numele subprogramului* pe care dorim să-l executăm. Subprogramele se stochează în spaţiul PROG al bazei de date cu ajutorul macrogeneratorului astfel :

```
%      RUN FN:M
:DEFPRO ⟨nume-subprogram⟩ [:CONTXT ...] :EXP
⟨model expansiune⟩
:FDEF ?
```

- NRX : zonă de memorie de tip cuvînt binar iniţializată cu o valoare egală cu numărul variabilelor SOCRATE de tip X_i utilizate ca parametrii (definite în cîmpul : CONTXT prin D X_i = ...) de apel ai programului precompilat.

Dacă subprogramele nu au variabile X_i ($i \neq 0$) ca parametrii atunci lui NRX i se atribuie valoarea 0 iar argumentele PX1, PX2, . . .,PX_x se omit.

• PX_x : zone de memorie de tip cuvînt binar destinate să definească un punct curent pe baza de date, unde $x \in [1, NPC]$. Aceste variabile au drept corespondent variabilele de tip X_i din SOCRATE.

• NRY : zonă de memorie de tip cuvînt binar inițializată cu o valoare egală cu numărul variabilelor SOCRATE de tip Y_i utilizate de subprogram ca parametrii (de apel sau rezultate).

Dacă subprogramele utilizează local (nu le folosește ca parametrii) sau nu utilizează variabilele de tip Y_i atunci lui NRY i se atribuie valoare 0 iar argumentele PY1, PY2, . . .,PY_y se omit.

• PY_y : zona de memorie de tip zecimal despachetat (cu semn) de 16 octeți. Aceste variabile au drept corespondent variabilele de tip Y_i din SOCRATE.

• NRZ : zonă de memorie de tip cuvînt binar inițializată cu o valoare egală cu numărul variabilelor de lucru SOCRATE de tip Z_i utilizate de subprogram ca parametrii. Dacă subprogramul utilizează local variabilele Z_i (sau nu utilizează acest tip de variabile) atunci lui NRZ i se atribuie valoarea 0 (zero) iar argumentele PZ1, PZ2, . . ., PZ_z se omit la apel.

• PZ_z : zona de memorie de tip alfanumeric de 30 octeți. Aceste argumente corespund variabilelor de lucru de tip Z_i ale sistemului SOCRATE.

Pentru utilizarea interfeței cu limbaje evoluate se fac precizările :

– în cazul în care se apelează mai multe subprograme SOCRATE în același timp care lucrează cu buffere formatate în diverse moduri aceste buffere trebuie să aibă aceeași lungime (subprogramele apelate în același timp sînt cele date în lista de parametrii ai unui apel la punctul de intrare SGBD) ;

– valorile argumentelor punctelor de intrare pot fi modificate pe parcursul prelucrării, această modificare trebuind să preceadă apelul, iar utilizatorul trebuie să prevadă secvențe logice pentru realizarea unui apel în concordanță cu valorile acestora. De exemplu, dacă argumentului NRX i se atribuie valoarea 1 atunci în sintaxa de apel a punctului de intrare SGBD trebuie să fie trecut un singur nume de zonă de tip PX_x ;

– numărul descrierilor de parametri de același tip și al valorilor asociate acestora este arbitrar, fiind obligatorie numai realizarea concordanței dintre sintaxa de apel și valorile atribuite acestora ;

– cu ajutorul interfeței este permisă utilizarea mai multor baze de date simultan cu condiția specificării acestui număr în parametrul UN al comenzii .OPTION.

Indiferent de limbajul gazdă ales ordinea operațiilor pe care le execută utilizatorul pe o bază de date oarecare este :

1) specificarea în zona FISIER a parametrilor de identificare a bazei de date dorite. Această specificare se realizează sau prin atribuirea unor valori cîmpurilor componente ale acestuia în corpul programului sau prin completarea lor prin dialog cu utilizatorul (de exemplu prin ordine ACCEPT în COBOL). Dacă parametrii sînt obținuți în urma unui dialog atunci utilizatorul trebuie să se asigure că acești parametrii sînt completați corect (dacă parametrul este mai scurt decît zona receptoare atunci pentru valori alfanumerice trebuie completat cu spațiu la dreapta iar pentru valori numerice cu zerouri nesemnificative la stînga). Recepționarea valorilor parametrilor poate fi efectuată în alte zone din memoria de lucru dar este absolut necesar trasferul lor corespunzător în zona care constituie parametru pentru un punct oarecare de intrare. Un transfer corezpunzător se referă atît la semnificație cît și la modul de

reprezentare internă (de exemplu valorile numerice se pot introduce de la consolă sub forma de reprezentare externă iar parametrul trebuie să conţină o valoare în binar, deci este necesară efectuarea unei conversii).

2) conectarea la baza de date dorită (LOGIN SOCRATE) prin apelul punctului de intrare **SOPEN** ;

3) efectuarea operaţiilor dorite pe baza de date deschisă apelînd programele SOCRATE prin intermediul punctului de intrare **SGBD**. Momentul şi modul de desfăşurare a acestor operaţii este stabilit de utilizator pe de o parte prin logica programului său iar pe de altă parte prin valorile atribuite parametrilor la momentul apelului ;

4) eventualele ordine de construire a unui (unor) punct(e) de control pentru baza sa de date prin apelul punctului de intrare **SSAVE**. Utilizarea acestui punct de intrare impune specificarea numelui şi suportului jurnalului (prin comenzile .ASSIGN Z, . . . şi .LABEL Z, . . .) şi alegerea tipului de jurnal dorit prin precizarea argumentului SF al comenzii .OPTION ;

5) închiderea bazei de date prin apelul punctului de intrare **SCLOSE**. O bază de date închisă devine indisponibilă pentru ordinele SGBD, SSAVE şi SCLOSE şi disponibilă pentru SOPEN şi STERM. Prin ordinul SCLOSE se consideră terminată o (mini) sesiunea pe baza de date implicată ;

6) terminarea sesiunii de lucru şi închiderea fişierului jurnal. Indiferent care este numărul de baze de date cu care se lucrează simultan acest ordin se dă o singură dată înaintea opririi definitive a execuţiei programului.

La execuţia lui STERM toate bazele de date cu care lucra programul trebuie să fie închise (cu ordinul SCLOSE adecvate).

Concluzionînd rezultă :

— efectuarea unei operaţii de tipul 2) sau 5) trebuie realizată o singură dată în cadrul unei (mini) sesiuni de lucru ;

— efectuarea unei operaţii de tipul 3) sau 4) se poate realiza ori de cîte ori doreşte utilizatorul ;

— efectuarea unei operaţii de tipul 6) trebuie realizată o singură dată pentru sesiunea de lucru (în fapt, această operaţie închide sesiunea de lucru similar unui LOGOUT) ;

— toate operaţiile enunţate pot fi precedate şi/sau urmate de diverse operaţii exprimate în comenzile limbajului gazdă şi conform logicii algoritmului de prelucrare stabilit de utilizator.

În V.1.6.R sintaxa punctului de intrare SGBD a fost modificată pentru a permite utilizarea variabilelor de tip W_1 ca parametrii, astfel :

SGBD ⟨FISIER, BUF, LGBUF, NRPGM, PGM1, PGM2, . . ., PGM_p,
NRX, PX1, PX2, . . ., PX_x,
NRY, PY1, PY2, . . ., PY_y,
NRW, PW1, PW2, . . ., PW_w,
NRZ, PZ1, PZ2, . . ., PZ_z⟩

• NRW : zonă de memorie de tip cuvînt binar iniţializată cu o ⟨*valoare*⟩ egală cu numărul de variabile de tip W_1 utilizate ca parametrii ;

• PW_1 : zone de memorie de tip zecimal împachetat reprezentate pe 12 octeţi. Aceste variabile au drept corespondent variabilele de tip W_1 din SOCRATE (*S9(15)V9(7)COMP-1*).

Sintaxa de apel şi descriere a parametrilor punctelor de intrare în limbajul COBOL

```
77      TIP           PIC X                      VALUE    '<tip>'.
*       <tip> :: = I/S/0
77      NPC           PIC S9(8)      COMP        VALUE    <valoare1>.
77      NCP           PIC S9(8)      COMP        VALUE    <valoare2>.
77      NCD           PIC S9(8)      COMP        VALUE    <valoare3>.
77      NBS           PIC S9(8)      COMP        VALUE    <valoare4>.
77      LGBUF         PIC S9(8)      COMP        VALUE    <valoare5>.
77      NRX           PIC S9(8)      COMP        VALUE    <x>   .
77      NRY           PIC S9(8)      COMP        VALUE    <y>   .
77      NRZ           PIC S9(8)      COMP        VALUE    <z>   .
77      NRPGM         PIC S9(8)      COMP        VALUE    <p>   .
* Descrierea variabilelor de tip PXx (X1)
   77  PX1      PIC   S9(8)      COMP.
   . . . . . . . . . . . . . . . . .
   77  PXx      PIC   S9(8)      COMP.
* Descriere variabile de tip PZz (Z1)
   77  PZ1      PIC     X(30).
   . . .
   77  PZz      PIC     X(30)
* Descriere variabile de tip PGMp
   77  PGM1     PIC     X(30)     VALUE    '<nume PGM1>'.
   . .   . . . . . . . . .
   77  PGMp     PIC     X(30)     VALUE    '<nume PGMp>'.
* Descriere variabile de tip PYy (Y1)
   01  PY1.
       05  FIL      PIC     X(6)      VALUE   LOW-VALUE.
       05  Y1       PIC     S9(18)    COMP-3.
   . . . . . . . . . . . . . . . . . .
   01  PYy
       05  FIL      PIC    X(6)       VALUE   LOW-VALUE.
       05  Yy       PIC    S9(18)     COMP-3.
* Descriere zona <FISIER>
   01  FISIER.
       05  FIL1     PIC       X(4)       VALUE SPACES.
       05  STRU     PIC       X(12)      '<nume-bază>' .
       05  UTI      PIC       X(18)      '<nume-util>' .
       05  AN       PIC       X(4)       '<cont-util>'.
       05  PW       PIC       X(8)       '<parola>'  .
       05  ERR
           10 CODER PIC S9(8) COMP.
           10 SUBRER        PIC X(30).
       05  FIL2             PIC X(58).
* Descriere buffer(e) de comunicaţie <BUF1>
01 BUF1.
```

⟨descriere conform structurii formalului logic asociat respectînd regulile de reprezentare a caracteristicilor⟩

Sintaxa de apel a punctelor de intrare :

1) deschiderea unei baze de date :
   CALL **SOPEN** USING FISIER, TIP, NPC, NCD, NBS [•]
2) închiderea unei baze de date :
   CALL **SCLOSE** USING FISIER [•]
3) crearea unui punct de control pe fişierul jurnal :
   CALL **SSAVE** USING FISIER [•]
4) terminarea sesiunii de lucru şi închiderea fişierului jurnal :
   CALL **STERM** [•]

Atenţie acest punct de intrare trebuie apelat imperativ înaintea execuţiei instrucţiunii STOP RUN indiferent dacă se lucrează sau nu cu fişier jurnal (altfel se editează mesajul de eroare P203).

5) realizarea accesului efectiv la baza de date :
   CALL **SGBD** USING FISIER, BUF, LGBUF,
   NRPGM, [PGM1, . . ., PGM_p]
   NRX, [PX1, . . ., FX_x]
   NRY, [PY1, . . ., PY_y]
   NRZ, [PZ1, . . ., PZ_z] [•]

La utilizarea variabilelor PY_y utilizatorul introduce şi extrage datele din cîmpul Y_y al variabilei.

Numărul zonelor de tip BUF sau structurile asociate unei zone BUF depind de numărul şi structura caracteristicilor de tip FORMAL utilizate de diversele subprograme. În cazul în care se apelează mai multe subprograme SOCRATE în acelaşi timp acestea trebuie să lucreze cu caracteristici FORMAL de aceeaşi lungime.

Numărul variabilelor de tip PX_x, PY_y şi PZ_x descrise în COBOL poate depăşi pe cel utilizat de subprograme, dar la apel vor fi furnizate ca argumente atîtea zone cîte indică variabila care dă numărul de variabile (NRX, NRZ sau NRY).

Sintaxa de apel şi descriere a parametrilor punctelor de intrare în limbajul – ASSIRIS

* Definirea punctelor de intrare externe

ENTREF SOPEN, SGBD, SSAVE, SCLOSE, STERM

* Definire şi iniţializare parametrii

TIP TEXT '⟨*tip*⟩'.
* ⟨*tip*⟩ : : = I/S/0
NPC DATA,4,4, ⟨*valoare*$_1$⟩
NCP DATA,4,4 ⟨*valoare*$_2$⟩
NCD DATA,4,4 ⟨*valoare*$_3$⟩
NBS DATA,4,4 ⟨*valoare*$_4$⟩
LGBUF DATA,4,4 ⟨*valoare*$_5$⟩
NRX DATA,4,4 ⟨*x*⟩
NRY DATA,4,4 ⟨*y*⟩
NRZ DATA,4,4 ⟨*z*⟩
* Descriere variabile PX_x (X_i)
PX1 RES,4 1
. . .
PX_x RES,4 1

```
* Descriere variabile PZ_z (Z_i)

PZ1        RES,4       8
...
PZ_z       RES,4       8

* Descriere variabile PY_y (Y_i)

                BOUND 8
PY1        RES,1       16
...
PY_y       RES,1       16

* Descriere buffer de comunicaţie

              BOUND    8
BUF        RES,1       <valoare_5>

* Descriere variabile PGM_p

NRPGM      DATA,4,4    <p>
PGM1       TEXT        '<nume PGM_1>'
.....
PGM_p      TEXT        '<nume PGM_p>'
* <nume PGM_i> se defineşte pe 30 caractere adăugînd
* eventual spaţii la dreapta dacă numele său are mai puţine
* caractere.
* Adăugarea de spaţii la dreapta este valabilă pentru orice cîmp care primeşte
* o valoare alfanumerică (se realizează rezervarea de spaţiu şi aliniere)
* Descriere zona <FISIER>

FISIER     RES,4       0
FIL1       TEXT        '     '/* spaţii
STRU       TEXT        '<nume-baza>'/* 12 caractere
UTI        TEXT        '<nume-uti>' /*  8 caractere
AN         TEXT        '<cont-uti>' /*  4 caractere
PW         TEXT        '<parola>' /*  8 caractere
ERR        RES,4       0
CODER      RES,4       1
SUBRER     RES,1       30
FIL2       RES,1       58

* Descrierea tabelelor cu adresele argumentelor punctelor
* de intrare
TABSOPEN RES,4         0
/* Parametrii punctului de intrare SOPEN

           DATA,4,4    FIŞIER
           DATA,4,4      TIP
           DATA,4,4    NPC
           DATA,4,4    NCD
           DATA,4,4    NBS
```

* Parametrii punctelor de intrare pentru SCLOSE şi SSAVE

```
TABCLSAV RES,4        0
         DATA,4,4     FISIER
```

* Parametrii pentru punctul de intrare ⟨SGBD⟩

```
TABSGBD  RES,4        0
         DATA,4,4     FISIER, BUF, LGBUF
         DATA,4,4     NPRGM
         [DATA,4,4    PGM1,...,PGMp]
         DATA,4,4     NRX
         [DATA,4,4    PX1,...,PXx]
         DATA,4,4     NRY
         [DATA,4,4    PY1,...,PYy]
         DATA,4,4     NRZ
         [DATA,4,4    PZ1,...,PZz]
```

Pentru punctele de intrare care necesită specificarea unor argumente la apel trebuiesc efectuate operaţiile :

– încărcarea în registrul general 2 (R2) a adresei tabelei din memorie care conţine adresele argumentelor ;
– apelul efectiv al punctului de intrare (a funcţiei dorite).

Sintaxa de apel a punctelor de intrare :

1) deschiderea unei baze de date :

```
LD4I,2      TABSOPEN
CALL,L      SOPEN
```

2) închiderea unei baze de date :

```
LD4I,2      TABCLSAV
CALL,L      SCLOSE
```

3) creerea unui punct de control pe fişierul jurnal :

```
LD4I,2      TABCLSAV
CALL,L      SSAVE
```

4) terminarea sesiunii de lucru şi închiderea fişierului jurnal :

```
CALL,L      STERM
```

Acest apel trebuie să preceadă oricărei decizii de terminare sau abandonare a acţiunii programului.

5) realizarea accesului efectiv la baza de date :

```
LD4I,2      TABSGBD
CALL,L      SGBD
```

Sintaxa de apel şi descriere a parametrilor punctelor de intrare în limbajul FORTRAN

```
⑤  ⑥  ⑦  /* coloanele din linia sursă */
C      Rezervarea zonelor de memorie destinate parametrilor
       INTEGER, FISIER, STRU, UTI, PW, AN, CODER, SUBRER, FIL1
       DIMENSION FISIER (32), STRU (3), UTI (2), PW (2), SUBRER (8)
```

```
C    Rezervare parametrii de tip PGM_p
     INTEGER PGM1,...,PGM_p
     DIMENSION PAGM1 (8), , PGM_p(8)
C    Rezervare parametrii de tip PY_y (Y_i)
     INTEGER PY_1,...,PY_y
     DIMENSION PY_1(4),...,PY_y(4)
C    Rezervare parametrii de tip PZ_z (Zi)
     INTEGER PZ_1,...,PZ_z
     DIMENSION PZ_1(8),...,PZ_z(8)
C    Rezervare parametrii de tip PX_x (X_i)
     INTEGER PX_1,...,PX_x
C    Structurare zona FISIER cu componentele sale
     EQUIVALENCE (FIL1, FISIER(1)), (STRU(1), FISIER(2)),
*                (UTI(1), FISIER(5)), (AN, FISIER(7)),
*                (PW(1), FISIER(8)), (CODER, FISIER(10)),
*                (SUBRER(1), FISIER(11))
C    Rezervarea bufferului de comunicaţie :
C    (Structura eventuală a bufferului va fi realizată în funcţie de caracteristica de
     tip formal asociată)
     DIMENSION BUF (lg) unde ⟨lg⟩ se calculează cu
```

formula : $\langle valoare_5\rangle/4 = q + r : \begin{cases} \text{dacă } r = 0 \text{ atunci } \langle lg\rangle = q \\ \text{dacă } r \neq 0 \text{ atunci } \langle lg\rangle = q + 1 \end{cases}$

⟨*valoare*$_5$⟩ reprezintă lungimea bufferului de comunicaţie

```
C    Atribuirea valorilor parametrilor de apel
     DATA TIP /' ⟨tip⟩ '/
C    ⟨tip⟩ : : = I/S/O
     DATA FIL1/'    '/ /* patru spaţii între apostrofi
     DATA STRU/' ⟨nume-baza⟩'/
     DATA UTI/' ⟨nume-uti⟩'/
     DATA AN/' ⟨cont-uti⟩'/
     DATA PW/' ⟨parola⟩'/
     DATA NPC, NCP, NCD, NBS/⟨valoare_1⟩ ⟨valoare_2⟩, ⟨valoare_3⟩, ⟨valoare_4⟩
     DATA LGBUF, NRX, NRY, NRZ, NRPGM/⟨valoare_5⟩, ⟨x⟩, ⟨y⟩, ⟨z⟩, ⟨p⟩/
C    Programele SOCRATE apelate
     DATA PGM1/' ⟨nume PGM_1⟩'/
     ...
     DATA PGM_p/' ⟨nume PGM_p⟩'/
```

Sintaxa de apel a punctelor de intrare :

1) deschiderea unei baze de date :
   CALL SOPEN (FISIER, TIP, NPC, NCP, NCD, NBS)
2) închiderea unei baze de date :
   CALL SOPEN (FISIER)
3) crearea unui punct de control pe fişierul jurnal :
   CALL SSAVE (FISIER)
4) terminarea sesiunii de lucru şi închiderea fişierului jurnal :
   CALL STERM

Această subrutină trebuie executată înaintea instrucţiunii STOP.

5) realizarea accesului efectiv la baza de date :

CALL SGBD (FISIER, BUF, LGBUF,
NRPGM, [$PGM_1, \ldots, PGM_p$]
NRX, [$PX_1, \ldots, PX_x$,]
NRY, [$PY_1, \ldots, PY_y$,]
NRZ [,$PZ_1, \ldots, PZ_z$])

Obţinerea programelor în format executabil

Pentru obţinerea programelor, în format executabil, care utilizează interfaţa cu limbaje evoluate se execută o link-editare a codului obiect al acestora cu un ansamblu de module, în format RBN, ale SGBD-SOCRATE. Programul utilizator devine „rădăcină" a programului executabil iar modulele SOCRATE apelate se reacoperă conform unui ordin .TREE (SIRIS 3) standard. Structura generală a modului de apel şi reacoperire a segmentelor RBN SOCRATE este prezentată în procedura LINKCOBO (anexa 2).

Această procedură pune la dispoziţia utilizatorului următorii parametrii formali :

— D : specifică adresa discului pe care se află montată biblioteca cu modulele RBN SOCRATE (BIBRBNSO) şi are valoarea implicită ADO ;

— TREE : numele programului obiect al utilizatorului. Deoarece programul devine un segment rădăcină acest nume trebuie să fie format din maxim 6 caractere. Dacă acest nume are mai puţin de 6 caractere va fi completat la dreapta cu un număr corespunzător de caractere %. Acest nume reprezintă pentru programele scrise în limbajul :

— COBOL : numele atribuit în PROGRAM-ID ;

— ASSIRIS : eticheta punctului de intrare în program (argumentul ordinului END) ;

— FORTRAN : argumentul ordinului .SEG care precede ordinul .COMPILE.

Dacă numele programului obiect este specificat în ordinul .COMPILE atunci acesta va fi utilizat în mod obligatoriu.

Numele atribuit programului executabil rezultat în cartela •LINK poate să nu aibă nici o legătură cu cel al programului în cod obiect.
Structura generală a JOB-ului de obţinere a unui program executabil utilizînd interfaţa cu limbaje evoluate este :

. COMPILE ⟨*nume compilator*⟩ [⟨*lista argumente*$_c$⟩] [*comentariu*]

{ cartele conţinînd codul sursă al programului

. XPROC LINKCOBO, MOD

. MOD &TREE : ⟨*nume program obiect*⟩ &

. ENDMOD

cartele pentru conectarea la :

{ — o bază de baze : .ASSIGN S..., .LABEL S... ;
— fişierul de manevră SOCRATE : .ASSIGN Q,..., .LABEL Q ;
— baza de date : .ASSIGN T... ;
— fişierul jurnal : .ASSIGN Z,...,. LABEL Z,... }

dacă nu se introduc aceste comenzi atunci se vor specifica înaintea lansării în execuţia programului

{ — fişierele utilizator : aceste fişiere pot avea ca suport
tipurile acceptate de versiunea utilizată

- LINK [⟨*lista argumente*$_e$⟩] — [⟨*ccmentariu*⟩]

Lansarea în execuţie se realizează conform modului de lansare, al oricărui program, sub controlul sistemului de operare SIRIS-3 sau HELIOS.

Pentru sintaxa ordinelor SIRIS-3 sau HELIOS şi a listei de argumente a acestora consultaţi specificaţiile limbajului de comandă al sistemului de operare SIRIS-3 sau HELIOS.

6. UTILIZAREA FUNCŢIUNILOR SGBD-SOCRATE

SGBD-SOCRATE pune la dispoziţia utilizatorilor un set de module tip monitor de baze de date ale căror funcţiuni permit :

— xxxGBDB : — generarea bazei de baze ;
— xxxGBAS : — generarea unei baze de date ;
— formatarea parţială sau totală a bazei de date ;
— schimbarea dimensiunii spaţiului real ;
— statistici parţiale sau totale asupra spaţiului sau spaţiilor bazei de date ;
— vidaj parţial sau total asupra spaţiilor sau subspaţiilor bazei de date ;
— xxxBTCH : — utilizarea funcţiunilor macrogeneratorului ;
— definirea structurii bazei de date ;
— utilizarea programelor de cereri (creare, modificare, exploatare) ;
— utilizarea funcţiunilor editorului de texte ;
— xxxMULC : — exploatarea în conversaţional — mod caracter — a bazei de date (sînt accesibile toate funcţiunile lui xxxBTCH) ;
— xxxMULM : — exploatarea în conversaţional — mod mesaj — a bazei de date (idem xxxMULC) ;
— xxxREST : — utilizarea funcţiunilor de refacere totală sau parţială a bazei de date în caz de incident ;
— utilizarea funcţiunilor de creare şi exploatare a fişierelor jurnal ;
— FERME : — permite închiderea, în sens SOCRATE şi SGF a unei benzi jurnal rămase deschisă după un incident hard sau soft. Acest modul este de tip procesor independent.

unde, xxx este :
— SOC pentru V.1.5 ;
— LINK pentru V.1.6.R.

Notă : realizatorii V 1.6.R au monitorizat funcţiunile ***MULC şi ***MULM în acelaşi modul (în sensul acceptării de către acelaşi modul atît a terminalelor conversaţionale „mod caracter" cît şi „mod mesaj").

Definirea unei proceduri generale de apel a modulelor SGBD SOCRATE

Pentru a explica şi exemplifica modul de apel şi execuţie a modulelor SGBD SOCRATE vom defini un fişier de comenzi sub formă de „procedură catalogată" cu parametrii formali (cu valori implicite) adaptabilă la diverse situaţii de execuţie. Această procedură va fi definită ca procedură în „stream" (sau va fi stocată în biblioteca Z%PROC a sistemului) pentru a simplifica sintaxa sa de apel.

Structura acestei proceduri este :

```
1    11                                                              72
●    XPROC PROCGEN,DEF
●    FETCH LN:BIBIMTSO,GN:&GN:1&,VN:&VN,&MODUL:SOCBTCH&,              *
     &SUPB:DVT:AD,VS:AD1AD1&
●    ASSIGN S,&SUPS:DVT:AD,VS:AD1AD1&       SUPORT PENTRU BAZA DE BAZE
●    ASSIGN Q,&SUPQ:DVT:AD, VS:AD1AD1&      SUPORTUL PENTRU MANEVRA
                                            SGBD
●    ASSIGN T,DV:&SUPBD:AD2+AD3&            SUPORTUL BAZEI DE DATE
                                            (MULTIVOLUM)
●    &JURNAL &ASSIGN Z,&SUPJ:DV:MT**&
●    &JURNAL&LABEL Z,FN:'&FNZ:NUME JURNAL%'
●    LABEL S,FN:'&FNS:BAZA DE BAZE&',AM:OFL
●    LABEL Q,FN:'&FNQ:MANEVRA SGBD&',AM:OFL
●    ENDPROC
```

unde :

⟨MODUL⟩ : numele modulului SOCRATE apelat ;

⟨SUPB⟩ : suportul bibliotecii BIBIMTSO care conţine modulele IMT-SOCRATE ;

⟨SUPS⟩ : suportul pe care se află ⟨*baza de baze*⟩ cu numele ⟨FNS⟩ ;

⟨SUPQ⟩ : suportul pe care se află fişierul de manevră SOCRATE cu numele ⟨FNQ⟩ ;

⟨JURNAL⟩ : atribuirea valorii '*-' permite renunţarea la lucrul cu fişier jurnal (liniile corespunzătoare sînt transformate în linii comentariu !) ;

⟨SUPBD⟩ : suportul/suporţii pe care se află baza de date al cărei nume se specifică la conectare (argumentul BN al comenzii %SOC). Dacă se omite din procedură această comandă atunci, după validarea cererii de conexiune la baza de date se va solicita, la consola sistemului, specificarea suportului (suporţilor) bazei astfel : *pp* ⟨*nume-baza*⟩ DV : operatorului revenindu-i sarcina specificării suportului (sau suporţilor) sub forma DK ** (sau $DK_i + DK_j + \ldots$), DK reprezentînd tipul real al suportului (MD, AD, BD, CD . . .). După validarea numelui logic al suportului este cerută adresa fizică a acestuia sub forma : *pp* ADRESSE DE + DK** : adresa ⟨cr⟩ (în cazul unei liste de suporţi se va cere adresa fiecăruia printr-un mesaj de această formă). *pp* este numărul partiţiei în care se lansează în execuţie modulul.

Rolul acestei proceduri este acela de a instala modulul în memoria operativă din memoria externă pe care se află şi de a-i furniza datele specifice fişierelor cu care lucrează acesta.

Pentru lansarea efectivă în execuţie, după apelul modulului sub forma :

```
●    XPROC PROCGEN [,MOD]
●    MOD &MODUL: ⟨nume-modul⟩ & [,& ⟨parametru⟩ : ⟨valoare⟩, ...]
●    MOD & ⟨parametru⟩: ⟨valoare⟩ [,...]
●    ENDMOD
```

(pentru modulul SOCBTCH, dacă ceilalţi parametrii convin, argumentele opţionale se omit), se vor plasa, în această ordine :

– comanda .OPTION cu valorile dorite pentru parametrii de execuţie ai modulului ;

– comanda, monitorului SIRIS 3, .RUN pentru lansarea efectivă în execuţie ;
– comanda % SOC pentru conectarea la baza de date dorită ;
– comenzi de apel al componentelor modulului apelat urmate eventual de linii sursă destinate analizei acestora ;
– comanda % LOGOUT pentru terminarea sesiunii de lucru.

Exemplu :

Apelul modulului SOCGBAS, fără fişier jurnal, se execută astfel :

```
●   XPROC PROCGEN,MOD
● . MOD &MODUL:SOCGBAS&, &JURNAL: *&
●   ENDMOD
●   OPTION CS'CF:90,DS:4096'
●   RUN TIME:999,NL:5000 AD:0 0
. . . . . . . . . . . . . . . . . .
```

Utilizarea comenzii OPTION

Rol : furnizarea unor parametrii de execuţie ai modulului SOCRATE apelat.

Această comandă este un ordin al sistemului de operare SIRIS 3 şi va fi tratată de către monitorul de înlănţuiri al acestuia (parametrii vor fi plasaţi în zona de opţiuni a partiţiei în care este lansat modulul SOCRATE şi vor fi trataţi de către acesta).

Sintaxa :

① ⑪
● ⟨*eticheta*⟨ OPTION CS' ⟨*lista de opţiuni*⟩' [⟨*comentariu*⟩]
unde prin 1 şi 11 am specificat coloana din linia sursă.

Specificaţii :
⟨*eticheta*⟩ : 1 –8 caractere alfa-numerice (respectă regulile de formare a etichetelor pentru comenzile SIRIS 3) ;
⟨*comentariu*⟩ : trebuie să fie precedat de cel puţin un spaţiu şi este opţional ;
⟨*lista de opţiuni*⟩ : : = ⟨CF⟩, ⟨DS⟩ [,⟨SF⟩] [,⟨TM⟩] [,⟨UN⟩] [,⟨CT⟩]
⟨CF⟩ : : = CF : n_1 – specifică numărul de cadre fişier necesare prelucrării. Acest număr poate fi superior celui declarat la generarea bazei de date.

Numărul de cadre declarat este limitat de :
– mărimea zonei de memorie liberă în partiţia de lansare ;
– numărul de intrări ale memoriei asociative (NBMASS) care este un parametru de generare a sistemului.

Mărimea unui cadru fişier este de 1 068 octeţi (1 pagină + informaţii de gestiune şi control) iar mărimea minimă a lui n_1 este de 3 (2 cadre pentru utilizator şi 1 cadrul pentru sistem). Calculul numărului de cadre reale alocate (N) se face cu formula :

$$N = \inf \left[n_1,\ \text{NBMASS} - 1,\ \frac{P}{1\,068} - 1 \right] \text{ unde :}$$

– n_1 : valoarea lui CF ;
– P : mărimea zonei libere în partiţie ;
– inf : valoarea minimă dintre valorile din listă.

Specificarea unui număr de cadre cît mai mare îmbunătăţeşte performanţele de execuţie a funcţiunilor modulului apelat.

DS : : = $DS : n_2$ rezervarea unui buffer de n_2 octeţi necesar desfăşurării operaţiilor de intrare-ieşire. Acest buffer este rezervat în memoria liberă (mărimea sa poate limita numărul de cadre fişier). Argumentul ⟨DS⟩ este obligatoriu iar modul de calcul a valorii sale n_2 este :

– cu utilizarea fişierelor jurnal : $n_2 = 2\,400$ octeţi ;

– cu utilizarea unui formal :

1) pentru fişiere de tip „intrare standard" : $n_2 = 80$ octeţi ;

2) pentru fişiere pe suport magnetic : $n_2 = (BFS) + (RCS) + 40$ unde valorile lui (BFS) şi (RCS) sînt valorile afectate cuvintelor cheie, corespunzătoare declarate în cartela % FILE ;

– cu utilizarea interfeţei cu limbaje evoluate : pentru fiecare utilizator se rezervă o zonă a cărei mărime minimă este de $(50 + p*3)$ cuvinte, unde p este numărul de puncte curente. Dacă u este numărul de utilizatori declaraţi la utilizarea simultană a sistemului (u baze de date utilizate simultan) atunci valoarea lui DS este : $(50 + p * 3)*u*4$ octeţi. În cazul în care la un apel se utilizează simultan mai multe funcţiuni care necesită rezervare în ⟨DS⟩ atunci valoarea lui n_2 se determină ca sumă a necesarului de memorie cerut de fiecare funcţiune în parte. Pentru o sesiune de lucru, valoarea argumentului DS trebuie să satisfacă cea mai mare cerere de spaţiu static.

⟨TM⟩ : : = TK : {Y | N} – argumentul precizează că se doreşte (Y) sau nu (N-implicit) activarea nivelurilor de urmărire a modului de desfăşurare a prelucrării ;

⟨UN⟩ : : = *UN* : *u* – argumentul este specific interfeţei cu limbaje evoluate şi precizează numărul maxim de baze de date care pot fi deschise simultan ;

⟨CT⟩ : : = CT : ({W | R | U},nn) – argumentul este rezervat modulelor multi şi permite precizarea criteriului de declanşare a constituirii unui punct de control (Checkpoint Triggering) ;

– *nn* – precizează numărul maxim de accesuri la baza de date între două CKPT succesive ;

Calculul numărului de accesuri se face :

– W : în scriere ;

– R : în citire ;

– U : în scriere şi citire.

Omiterea argumentului CT invalidează declanşarea constituirii punctelor de control.

⟨SF⟩ : : = SF : ([JNLA] [, JNLB]) – cererea de constituire a jurnalelor A şi B (SF : Security File). Argumentele JNLA şi JNLB sînt poziţionale iar cererile de constituire pot fi :

– SF : (JNLA) : numai jurnalul A ;

– SF : (,JNLB) ; numai jurnalul B ;

– SF : (JNLA, JNLB) : ambele jurnale.

Exemplu :

OPTION CS'CF:10,DS:2400,TM:Y' solicită

– rezervarea a 10 cadre fişier ;

– rezervarea unei zone statice de 2 400 octeţi ;

– activarea nivelurilor de urmărire a prelucrării.

Limbajul de comandă

Comunicarea cu sistemul SOCRATE se realizează cu ajutorul unui limbaj de comandă prin care utilizatorul indică :

– funcţia de executat ;

– argumentele necesare execuţiei acestei funcţii.

Alfabetul limbajului de comandă este acelaşi cu cel descris în capitolul 2.

Structura limbajului de comandă este :

⟨limbaj de comandă⟩ : : = *⟨etichetă⟩* *⟨cod comandă⟩* *⟨cuvînt cheie⟩* *⟨identificator⟩*

⟨eticheta⟩ : 1–8 caractere alfanumerice ;

⟨cod comandă⟩ : 1–8 caractere alfanumerice ;

⟨cuvînt cheie⟩ : mnemonică, în engleză, formată din două caractere alfanumerice urmată de caracterul „:" care permite atribuirea unei valori la mnemonică. Detectarea valorii se face luînd toate caracterele cuprinse între „:" şi primul caracter spaţiu sau "," ;

⟨identificatori⟩ : şiruri de 1–8 caractere alfanumerice care nu conţin nici spaţiu, nici virgulă.

Tipul comenzilor diferă ca structură şi mod de introducere în sistem în funcţie de timpul de prelucrare ales de utilizator („batch processing" sau conversaţional)

Limbajul de comandă în prelucrarea „batch processing" (SOCBTCH)

Selecţia funcţiunii dorite (comenzii) şi furnizarea argumentelor necesar execuţiei acestei funcţiuni se realizează cu ajutorul liniilor de comandă SOCRATE care au următoarea sintaxă generală de descriere :

%[*⟨eticheta⟩*] – *⟨comanda⟩* – *⟨lista de argumente⟩* – [*⟨comentariu⟩*]

⟨lista de argumente⟩ : : = *⟨argument⟩*/*⟨argument⟩*, *⟨lista de argumente⟩*

⟨argument⟩ : : = *⟨mnemonică⟩*: *⟨valoare⟩*

În această structură generală există coloane cu destinaţie strictă, ele fiind prezentate în tabela 6.1.

Tabelul 6.1. Structura liniei de comandă SOCRATE

Coloana	Valoare	Explicaţii
1	%	identificatorul liniei de comandă SOCRATE
2–9	⟨eticheta⟩	opţional, comanda poate fi etichetată
10	spaţiu	
11	⟨comanda⟩	numărul de coloane ocupat de comandă diferă în funcţie de numele acesteia. Indiferent care este lungimea se va lăsa un spaţiu liber după care continuă lista de argumente
72	*/Ƀ	prezenţa unui * în această coloană anunţă sistemul că lista de argumente se continuă pe linia următoare : Ƀ– linie singulară
20	continuare ⟨lista de argumente⟩	linia care urmează unei comenzi care a anunţat continuarea este analizată începînd cu coloana 20

În cazul în care ⟨*lista de argumente*⟩ nu încape în linie (ultimul argument urmat de „," și eventual de spații pînă în col-71) se trece „*" în coloana 72, celelalte argumente continuîndu-se pe linia următoare începînd cu coloana 20 (coloana 1 conține caracterul „%").

Comanda SOC

Rol : efectuarea conexiunii la o bază de date.
Sintaxa :
% [⟨*eticheta*⟩] SOC BN : ⟨*nume-baza*⟩, PN : ⟨*nume-uti*⟩, AN : ⟨*cont-uti*⟩, PW : ⟨*parola*⟩
BN : Base Name
⟨*nume-baza*⟩ : numele bazei de date la care dorim să ne conectăm ;
PN : Programer Name ;
⟨*nume-uti*⟩ : numărul său de cont ;
AN : A count Name
⟨*cont-uti*⟩ : numărul său de cont ;
PW : Pass Word
⟨*parola*⟩ : parola utilizatorului.
Specificații :
Argumentul PN, AN și PW trebuie să fie introdus, de către administrator, în spațiul asociat bazei de date (din baza de baze care gestionează ⟨*nume-baza*⟩) cu ajutorul comenzilor de tip UTI sau a macroinstrucțiunilor sistemului de gestiune a datelor din baza de baze. Dacă în urma verificării identității acestor valori cu cele asociate pentru ⟨*nume-bază*⟩ în baza de baze se detectează inadvertențe atunci cererea de conexiune va fi respinsă cu editarea unui mesaj de eroare adecvat.

Această comandă este utilizată în același scop și de modulele SOCGBAS și SOCREST.

Exemple :

```
%    SOC BN:BDDPERS,PN:ASE,PW:ASE AN:1555
% CONECT SOC BN:BDDPERS,PN:AVRAM,PW:SOC,AN:0001
```

Comanda EVAL

Rol : evaluarea performanțelor de execuție a procesoarelor și programelor apelate sub controlul său.
Sintaxa :
% [eticheta] — EVAL —[⟨*comentariu*⟩]
Specificații :

Acest ordin permite evaluarea performanțelor de execuție a programelor și procesoarelor, evaluare utilizabilă în principal pentru optimizarea programelor. Rezultatul evaluării este imprimat, pentru fiecare acțiune în parte, sub forma :

T : *xx*/*yy*/*zz*
D : *dd*/*pp*
unde :
— *xx* — timpul UC consumat ;
— *yy* — timpul aparent (*yy* = *xx* + *zz*) ;

— zz — timpul necesar operațiilor de intrare/ieșire ;
— pp — numărul de accesuri la paginare ;
— dd — numărul de accesuri la spațiul fizic (citiri și scrieri efective de pagini).

Controlul comenzii se extinde asupra tuturor procesoarelor și programelor executate între ea și comanda :

% NOEV sau % LOGOUT

Exemple :

% EVAL
% EVAL EVALUARE PERFORMANTE

Comanda NOEV

Rol : inhibarea procesului de evaluare a performanțelor (anularea efectului comenzii EVAL).

Acest mod de funcționare este implicit.

Sintaxa :

% [⟨*eticheta*⟩] NOEV [⟨*comentariu*⟩]

Comanda ACTI

Rol : activarea opțiunilor speciale de execuție a modulelor.

Sintaxa :

% [⟨*eticheta*⟩] ACTI TB :** [⟨*comentariu*⟩]

Specificații :

În funcție de valoarea argumentului TB și procesorul SOCRATE apelat această comandă activează anumite opțiuni de execuție ale procesorului. Este permisă utilizarea în secvență a acestor comenzi (unele opțiuni solicită chiar utilizarea mai multor ordine ACTI). Un ordin ACTI este pus în factor pentru toate procesoarele apelate sub controlul său (poate fi inhibat cu ajutorul unei comenzi % DACTI sau automat la % LOGOUT) iar acțiunea sa este simultană cu a tuturor celorlalte ordine ACTI active. Majoritatea valorilor (nivelurilor) ** sînt rezervate administratorului bazei de date și permit acestuia să obțină vidaje ale registrelor generale și zonelor de memorie tratate de procesoarele modulului apelat, vidaje necesare punerii la punct a modulului în cazul efectuării unor modificări ale acestuia.

Valorile lui ** puse la dispoziția utilizatorului și semnificația acțiunii declanșate de acestea sînt :

1) ** = 49 :

— pentru procesorul D (definire) permite adăugarea la o structură existentă (structura începe cu D în locul lui DEBUT) sau redefinirea integrală a structurii (structura începe cu DEBUT) ;

— pentru procesorul R (compilatorul LMD sau cereri) permite obținerea codului intermediar ;

2) ** = 44 și ** = 64 — (utilizate împreună) permit obținerea unui tablou care conține codul intern al structurii bazei de date.

3) ** = 78 — înhibarea editării mesajelor de eroare pentru erorile detectate la validarea standard aplicată valorilor atribuite caracteristicilor definite în structura bazei de date.

4) ** = 43 – inhibarea editării standard a tabloului rezumat editat la definirea structurii sau la adăugarea la structură.

Exemplu :

```
%     SOC BN:BDDPERS,PN:ASE PW:ASE,AN:1555
%     EVAL EVALUARE PERFORMANTE
%     ACTI TB:49 ADAUGARE LA STRUCTURA
%     ACTI TB:64 OBTINERE
%     ACTI TB:44    COD INTERN
%     RUN FN:D APEL COMPILATOR  LDD
D /* ADAUGARE LA STRUCTURA ( DEBUT PENTRU ADAUGARE )
FORMAL CARTELA
  DEBUT
    IDENT   MOT 3
    BL-1    MOT 1
    NUME    MOT 15
    BL-2    MOT 1
    PRENUME MOT  15
    BL-3    MOT 1
    VIRSTA DILATE 3
  FIN
FIN
?
```

Comanda DACTI

Rol : dezactivarea opţiunilor speciale de execuţie.
Sintaxa :
% [⟨*eticheta*⟩] DACTI TB : ** [⟨*comentariu*⟩]
Specificaţii :

Opţiunea ** specificată este inhibată. La lansarea şi terminarea unei sesiuni de lucru toate opţiunile ** sînt inhibate în mod automat.

Comanda SAVE

Rol : specificarea cererii de constituire a unui punct de control.
Sintaxa :
% [⟨*eticheta*⟩] SAVE [⟨*comentariu*⟩]
Specificaţii :

Constituirea punctului de control se realizează numai dacă sesiunea de lucru a fost lansată cu „fişier jurnal" (argumentul SF al comenzii •OPTION).

Comenzi necesare descrierii fişierelor secvenţiale pe suport magnetic

a) Fişiere pe benzi magnetice

Pentru a lucra cu fişiere pe bandă magnetică este necesară următoarea secvenţă de comenzi, pentru fiecare fişier în parte :

1) % [*eticheta*⟩] ATTACH MT**
2) % [⟨*eticheta*⟩] ASSIGN DV : MTyy
3) % [⟨*eticheta*⟩] LABEL FN :' ⟨*nume-fişier*⟩' [,⟨arg_e⟩]

4) % [⟨*eticheta*⟩] FILE PRM : {INP/OUT}, RCS : *r*, BFS : *b*, RCF : {FIX/VAR/UND}[, ⟨arg_1⟩]

Comanda 1) este prima comandă care se dă pentru un fişier şi permite ataşarea fişierului logic** la SOCRATE. Numărul** trebuie citat ca atare pentru toate cererile de intrare/ieşire efectuate cu articolele acestui fişier (ordinele LIRE/ECRIRE).

Comanda 2) permite asocierea numărului logic SOCRATE (**) la un dispozitiv logic SIRIS 3 (HELIOS), căruia, la execuţie i se va ataşa adresa fizică efectivă a perifericului pe care este montată rola care conţine fişierul.
Pentru comenzile 3) şi 4) am prezentat numai argumentele obligatorii. Pentru aceste comenzi sînt valabile toate argumentele comenzilor SIRIS 3 sau HELIOS corespondente, pentru lucrul cu fişiere pe suport magnetic (de exemplu OFO, CFO, pentru tratarea volumelor mutifişier sau adăugarea la un fişier ; GN, VN pentru ⟨*nume-fişier*⟩ etc.).

Semnificaţia valorilor parametrilor specificaţi este :
INP : intrare (fişierul va fi utilizat numai în citire) ;
OUT : ieşire (fişierul va fi utilizat numai în scriere) ;
r : mărimea în octeţi a articolului logic ;
b : mărimea în octeţi a articolului fizic (blocului).

Relaţia care există între cei doi parametrii se exprimă cu formula (înregistrări cu format fix) :

$b = (r + 1) * fb + 8$, unde *fb* reprezintă factorul de blocare (număr de înregistrări logice în înregistrarea fizică).
FIX : fişierul conţine înregistrări cu format fix ;
VAR : fişierul conţine înregistrări cu format variabil ;
UND : fişierul conţine înregistrări cu format nedefinit.
⟨*nume-fişier*⟩ : numele fişierului.
Ordinea comenzilor 2) – 4) după comanda 1) este aleatoare. Pentru fiecare fişier, tratat de un program, se specifică o secvenţă de acest gen înaintea lansării în execuţie a procesorului (comanda % RUN) sub controlul căruia funcţionează programul. Asocierea unui fişier la un program nu este pusă în factor pentru celelalte procesoare apelate, în sesiunea de lucru, după acesta.

Pentru a vă forma o imagine asupra modului de utilizare a acestor comenzi urmăriţi exemplul dat în capitolul 4.

Fişierele şi volumele acestora pot fi de forma :
– mono-fişier : mono-volum ;
– mono-fişier : multi-volum ;
– multi-fişier : mono-volum ;
– multi-fişier : multi-volum.
Poziţionarea la un fişier aflat pe un volum multi-fişier poate fi efectuată sau cu ajutorul utilitarelor sistemului de operare (exemplu POSFICH) sau cu programele SOCRATE scrise în LMD, utilizînd evident combinaţii adecvate pentru parametrii OFO şi CFO ai comenzii % FILE. Numărul maxim de fişiere tratate simultan de către un program este 5.
b) Fişiere pe disc magnetic

1) versiunea SOCBTCH a C.Ce al C.S.P. admite utilizarea a două fişiere pe disc magnetic. Pentru a ataşa la SOCRATE unul din aceste fişiere se utilizează secvenţa de comenzi descrisă la punctul a) cu următoarele modificări :

— comanda ATTACH este dată astfel :
% [⟨*eticheta*⟩] ATTACH ADF*, unde * : : = 0/1 ;
— ⟨arg_e⟩ şi ⟨arg_f⟩ vor fi selecţionate din lista de argumente pentru tratarea fişierelor secvenţiale pe disc magnetic.

2) V 1.6.R admite ca suport pentru fişiere secvenţiale în afara benzii magnetice şi discul magnetic. Mai mult, pentru fiecare este posibil lucrul sub controlul AVR şi mărirea performanţelor de acces la înregistrări prin posibilitatea utilizării a două buffere pentru operaţiile de citire-scriere. Numele simbolic al fişierului pe disc rămîne sintactic acelaşi ca la banda magnetică. Pentru lucrul cu fişiere pe disc magnetic se utilizează secvenţa de comenzi :

$$\begin{array}{l}\%\ \text{ATTACH MTxx} \\ \%\ \textit{LABEL}\ \text{FN : '}\langle \textit{nume fişier} \rangle\text{'}\end{array}\left[\text{,AM :}\left\{\begin{matrix}\text{ANY}\\ \text{OFL}\\ \text{SPC}\end{matrix}\right\}\text{, SZ : . . .}\right]$$

$$\%\ \text{FILE DVT :}\left\{\begin{matrix}\text{MT}\\ \text{MD}\\ \text{AD}\\ \text{BD}\end{matrix}\right\}\text{, PRM :}\left\{\begin{matrix}\text{INP}\\ \text{OUT}\\ \text{UND}\end{matrix}\right\}\text{, BFS : }n_1\text{, RCS : }n_2\text{, BFN :}\left\{\begin{matrix}1\\ 2\end{matrix}\right\}$$

$$\%\ \text{ASSIGN}\ \left\{\begin{matrix}\text{DV :}\\ \text{DVT : }t\text{, VS : VVVVVV}\end{matrix}\right\}$$

xx — numărul logic atribuit discului ;
⟨*nume fişier*⟩ — numele fişierului.

Comanda % LABEL admite parametrii comenzii LABEL a monitorului SIRIS 2/3 sau HELIOS.

Parametrii comenzii % FILE au aceiaşi semnificaţie ca în versiunea V 1.5, cu deosebire că la specificarea argumentului BFN : 2 calculul zonei necesare, pentru acest fişier, în parametrul DS (comanda OPTION) se face cu formula : 2 * (BFS) + + (RCS) + 8, unde (BFS) şi (RCS) sînt valorile atribuite argumentelor respective :

⟨*t*⟩ — tipul suportului (⟨*t*⟩ : : = MT/RD/MD/AD/BD) ;
VVVVVV — volumul serial al discului/benzi.

Atît pentru fişierele pe disc cît şi pentru cele pe bandă magnetică s-a introdus posibilitatea închiderii acestora şi a eliberării volumelor şi bufferelor ataşate lor prin comanda :

% [⟨*eticheta*⟩] RELEASE MTxx

Fişierele aflate în citire sînt închise în mod automat, la detectarea unui ⟨FILE MARK⟩, iar variabilele X_1, ataşate formalului în citire, sînt puse la valoarea U, eveniment detectabil la nivelul programelor SOCRATE prin testarea variabilelor X_1.

Comanda RUN

Rol : lansarea în execuţie a procesoarelor SGBD-SOCRATE.
Sintaxa :
% [⟨*eticheta*⟩] RUN FN : ⟨*procesor*⟩ [IM : {C/$}] [,{*LST*/NLT}]
Specificaţii :
— argumentele subliniate sînt implicite ;
FN : permite specificarea procesorului dorit ;
⟨*procesor*⟩ : : = E/D/C/M/R/O
E : editorul de texte (cap. 5) ;
D : compilatorul LDD (definirea structurii — cap. 3) ;

C : creere totală a bazei de date (cap. 4) ;
M : macrogeneratorul (cap. 5) ;
R sau O : compilatorul LMD (cap. 4).

Pentru aceste procesoare, care au comenzi specifice, au fost indicate, în paranteză, paragrafele sau capitolele în care sînt descrise aceste comenzi.

IM : defineşte suportul de pe care se vor introduce comenzile (datele) supuse analizei procesorului ;

– C: datele sînt preluate de pe dispozitivul de intrare standard al partiţiei (*1);

– $ datele sînt preluate din spaţiul virtual al utilizatorului. Aceste date trebuiesc introduse în prealabil, în acest spaţiu, cu ajutorul „editorului de texte" (§ 5 – fişier curent).

LST : liniile sursă vor fi listate pe dispozitivul de ieşire standard al partiţiei (*2) ;
NLT : nu se listează liniile sursă.

Comanda LOGOUT

Rol : închiderea sesiunii de lucru SOCRATE.
Sintaxa :
% [⟨*eticheta*⟩] LOGOUT [⟨*comentariu*⟩]
Specificaţii

Comanda provoacă închiderea în sens SOCRATE a bazei de date (eventualele pagini modificate care nu au fost scrise pe disc sînt scrise corect), eliberarea resurselor ocupate, închiderea fişierelor jurnal şi a fişierelor utilizator.

Această comandă este utilizată de toate modulele apelabile în „batch processing" (xxxBTCH, xxxGBDB, xxxGBAS, xxxREST).

Limbajul de comandă în prelucrarea conversaţională (xxxMULC, xxxMULM)

Lansarea unuia din cele două module conversaţionale este semnalată la consola sistemului de calcul prin mesajul :

MULTITASKING EN SERVICE

În acest moment, operatorul are posibilitatea să activeze toate sau anumite linii de transmisie prin comanda :

OPI INP [⟨*p*⟩] (NL)

$$\langle p\rangle \text{ SOCRATE xx ACT}\left\{\begin{matrix}\text{ALL}\\ \text{TLxx}\\ \text{TTxx}\end{matrix}\right\}\langle cr\rangle\text{, unde :}$$

– *p* : numărul partiţiei în care este lansat modulul SOCRATE ;
– ALL : activarea tuturor liniilor ;
– TL*xx* : activarea unei linii în mod mesaj (xxMULM) ;
– TT*xx* : activarea unei linii în mod caracter (xxxMULC) ;
– *xx* : numărul simbolic al liniei ;
– *cr* : tasta retur de car ;

Caracterele subliniate se tastează de către operator iar între cuvintele tastate se lasă un singur spaţiu. Adresa fizică a liniei pe care dorim să o activăm va fi cerută prin :

$$\langle p\rangle \text{ ADRESSE DE} + \begin{Bmatrix}\text{TL}\\ \text{TT}\end{Bmatrix} xx :$$

După conectarea la linie şi activarea acesteia, operatorul are la dispoziţie un set de comenzi prin care poate controla liniile şi posturile cuplate la acestea.

Comenzi la dispoziţia operatorului central

Prin noţiunea de operator central vom înţelege un operator aflat la consola sau una din consolele (HELIOS) sistemului de calcul.

Dialogul între operatorul central şi sistemul SOCRATE se desfăşoară sub forma :
OPI INP [*p*] K ⟨cr⟩
p SOCRATE ** ⟨*comanda*⟩ ⟨cr⟩
⟨*comanda*⟩ : : = TERM/ABOR[T]/ENDSES/MDCT/CKPT/KILL/CLOSE/MAIL/MSG ALL/ACTI/DACT

TERM : sfîrşit SOCRATE ;
ABORT : sfîrşit SOCRATE + vidaj ;
ENDSES : sfîrşit constituire fişier jurnal pentru o bază ;
MDCT : modificarea perioadei de „time-sharing" ;
CKPT : cerere de constituire punct de control ;
KILL : forţare LOGOUT la terminal (post) ;
CLOSE : închiderea unei linii ;
MAIL : modificarea mesajului zilei (mesaj curent) ;
MSG ALL : transmiterea unui mesaj spre toate posturile ;
ACTI : activarea nivelului de urmărire a prelucrării ;
DACT : dezactivare nivel de urmărire a prelucrării.

Comenzile TERM şi ABORT salvează bufferele şi închid bazele active înaintea opririi prelucrării.

Sintaxa comenzilor prezentate este :

- MDCT ($\begin{Bmatrix}\text{W}\\ \text{R}\\ \text{U}\end{Bmatrix}$, nn) unde :
  - nn : numărul maxim de accese efectuate la bazele active între două puncte de control (CKPT) ;
  - W : contorizarea se face pentru scrieri ;
  - R : contorizarea se face pentru citiri ;
  - U : se contorizează ambele tipuri de acces (citire/scriere).
- ENDSES $\begin{Bmatrix}\text{ALL}\\ \langle\textit{nume-bază}\rangle\end{Bmatrix}$ unde :
  - ALL : toţi utilizatorii activi trebuie să tasteze LOGOUT (se va constitui un singur punct de control pentru toate bazele de date active).
  - ⟨*nume-bază*⟩ : utilizatorul (utilizatorii) bazei de date specificate trebuie să execute procedura de LOGOUT.

- KILL $\left\{\begin{matrix}\text{TLxx}\\ \text{TTxx}\end{matrix}\ \langle n\rangle\right\}$

— comanda forțează LOGUT pentru utilizatorul postului iar linia rămîne deschisă ;

— ⟨*n*⟩ : numărul de ordine al postului în lista de selecție (mod mesaj).

- CLOSE $\left\{\begin{matrix}\text{TLxx}\\ \text{TTxx}\end{matrix}\right\}$ — provoacă închiderea liniei după forțate LOGOUT a utilizatorului (utilizatorilor) liniei. În cazul în care programul unității de schimb este activ închiderea nu este efectivă decît după apariția mesajului de time-out :

FERME ETAT E/S 04 xxxx

- MAIL — permite schimbarea mesajului standard (RIEN A SIGNALER) cu mesajul tastat pe următoarea linie a consolei. Acest mesaj va fi transmis utilizatorilor care efectuează LOGIN sau la cererea expresă a operatorului unui post.
- MSG ALL ⟨*mesaj*⟩ — provoacă transmiterea mesajului la toți utilizatorii activi ;
- ACTI
- DACTI

} — permit activarea/dezactivarea nivelurilor de urmărire a prelucrării. ACTI presupune existența unei imprimante la partiția în care se lansează SOCRATE.

În cazul prelucrării în mod mesaj (SOCMULM), operatorul central mai dispune de următoarele comenzi :

- INIT TLxx *yyy* — deschiderea postului cu numărul *yyy* în lista de selecție a liniei *xx* ;
- CLR n — poziționarea automată a cursorului pe linia *n* (liniile care urmează sînt șterse). Dacă N este numărul de linii ale ecranului numerotate 0–N–1) atunci $0 \leqslant n \leqslant (N-2)$;
- COPY — copierea automată a ecranului, la sfîrșit de ecran, pe imprimantă (imprimantă de recopiere a ecranului — „hard copy") ;
- NOCOPY — anulează efectul comenzii COPY. Această comandă este implicită la lansare.

Eventualele erori de transmisie sînt semnalate la consola calculatorului prin :

— *mod caracter* :

⟨*mesaj*⟩ S/R ERR ⟨*sir hexa*⟩

unde ⟨*șir hexa*⟩ : starea programului și rezultatul operației de intrare/ieșire ;

— *mod mesaj* :

⟨*mesaj*⟩ LN : TLoi TE : xx

unde LN : TLoi : numele simbolic al terminalului ;

TE : xx : numărul terminalului în lista de selecție

⟨*mesaj*⟩ poate fi :

ER SGT E/S REG12 = xxxxxxxx : macro SENS/RECEIVE nu este lansată ;

ER SGT TRAD REG12 = xxxxxxxx : erori pe macro TRANSLATE ;

ER SGT ETAT E/S = xxxxxxxx : erori de transmisie semn ;

FERME ETAT E/S = xxxxxxxx : eroare de transmisie la închiderea postului.

Comenzi la dispoziția utilizatorului

Un terminal conversațional (post, terminal) afectat lui SOCRATE poate fi în una din stările :

— *Ieșire* : SOCRATE trimite un mesaj la terminal, mesaj care va fi afișat la acesta ;

— *intrare* : utilizatorul trimite un mesaj la calculator. Această stare este semnalată printr-un anumit caracter afişat la terminal sau printr-o întrebare explicită a lui SOCRATE ;

— *supraveghere* : terminalul aşteaptă tastarea caracterului special „atenţie" (C1N sau tasta BREAK).

Pentru lucru la terminal, operatorul dispune de un set de comenzi ierarhizat astfel :

— comenzi de primul nivel : permit interacţiunea operator-SOCRATE ;

— comenzi specifice unui procesor : permit activarea diverslor funcţiuni ale unui procesor apelat cu ajutorul unei comenzi de primul nivel.

Comenzile de primul nivel se împart în două categorii :

1) comanda iniţială :
LOGIN — iniţializarea lucrului cu o bază de date ;

2) comenzi curente :
GO — începutul unei tranzacţii ;
RESTART — reluarea unei tranzacţii întrerupte prin tastarea caracterului „atenţie" ;
LOGOUT — sfîrşitul lucrului cu o bază de date ;
NUMBER — obţinerea numărului utilizatorilor simultan activi ai unei baze de date ;
CLOCK — obţinerea orei sistemului de calcul ;
MAIL — obţinerea mesajului zilei ;
MESSOP — transmiterea unui mesaj la consola operatorului central ;
PAUSE — construirea unui punct de control ;
EVAL — activarea procedurilor de evaluare a performanţelor ;
NOEVAL — dezactivare EVAL ;
ACTI — activarea procedurilor de urmărire a prelucrării ;
DACTI — dezactivare ACTI.

Momentul utilizării acestor comenzi este arătat în figura 6.1.

La activarea postului de către operatorul central, sistemul trimite automat, la toate terminalele activate la care s-a tastat caracterul „atenţie", mesajul SOCRATE A VOTRE SERVICE şi caracterul (promptul) care permite introducerea unei comenzi de primul nivel.

Prezentăm în continuare sintaxa comenzilor de primul nivel :

1) **Comanda LOG[IN]** : cerere de conectare la o bază de date :

Sintaxa :

$LO[GIN] ⟨cr⟩
ACC.NUMBER : ⟨*cont-uti*⟩ ⟨cr⟩
NAME : ⟨*nume-uti*⟩ ⟨cr⟩
BASE : ⟨*nume-bază*⟩ ⟨cr⟩
MOT DE PASSE : ⟨*parola*⟩ ⟨cr⟩

Caracterele subliniate se tastează de către operator, celelalte sînt emise de SOCRATE.

⟨*cont-uti*⟩ : contul utilizatorului ;
⟨*nume-uti*⟩ : numele utilizatorului ;
⟨*nume-bază*⟩ : numele bazei de date cu care dorim să ne conectăm ;
⟨*parola*⟩ : parola utilizatorului.

Această comandă este simulată în „batch processing" prin comanda % SOC... .

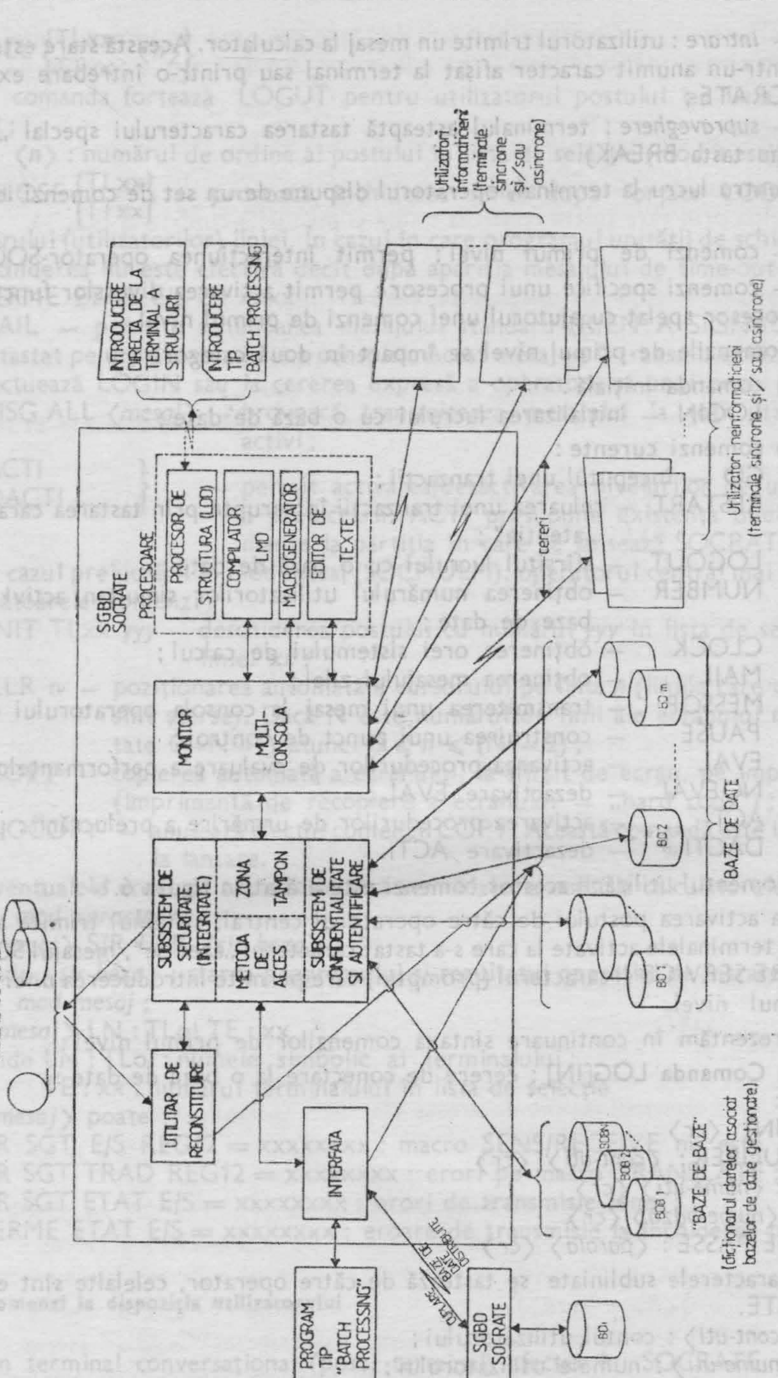


Fig. 6.1. Schema funcţională

Sistemul verifică validitatea parametrilor (prin compararea lor cu cei introduşi în baza de baze) şi în caz de eroare emite unul din mesajele :

MAX. UTIL : a fost atins numărul maxim de utilizatori simultani (parametru dat la generarea sistemului SOCRATE).

MAX. BASE : indică faptul că s-a depăşit numărul de baze simultan active (în SOCRATE V.1.5 pot fi deschise simultan 20 de baze de date, în regim de lucru conversaţional) ;

INCORRECT : ansamblul parametrilor nu este omogen (de exemplu utilizatorul indicat de ⟨*cont-uti*⟩ şi ⟨*nume-uti*⟩ nu are acces la baza ⟨*nume-bază*⟩);

ERR COM : parametrul furnizat este vid sau nu respectă formatul prevăzut (număr de caractere tastate mai mare decît cel prevăzut pentru parametru).

BASE INCONNUE : numele bazei de date este eronat. Dacă primii trei parametri sînt corecţi atunci sistemul cere tastarea parolei utilizatorului (după ce cere, eventual, adresa dispozitivului pe care se află montată baza de date la operatorul central) căreia îi verifică validitatea.

Dacă cuvîntul ⟨*parolă*⟩ este eronată se emite mesajul INCORRECT şi terminalul este adus în starea iniţială. Dacă toţi parametrii sînt corecţi, atunci SOCRATE afişează la terminal mesajul zilei (RIEN A SIGNALER) şi promptul **$** care permite tastarea comenzilor de primul nivel din categoria a doua.

2) **Comanda GO** : apelarea unui procesor SOCRATE.

Sintaxa :

$GO [⟨*procesor*⟩] ⟨cr⟩

⟨*procesor*⟩ : : = E/D/C/M/R/O

– E : editorul de texte ;
  D : compilatorul limbajului de definire ;
– C : procesorul de creere standard ;
– M : macrogeneratorul ;
– R, O : interpretorul limbajului de manipulare (apelul acestui procesor este implicit).

Procesorul apelat ia controlul intrărilor-ieşirilor la terminal şi pune la dispoziţia utilizatorului setul său propriu de comenzi.

3) **Comanda RESTART** : reluarea unei tranzacţii întreruptă prin tastarea caracterului „atenţie".

Sintaxa :

$REST[ART] ⟨cr⟩

Tranzacţia este reluată de la adresa la care a fost întreruptă.

4) **Comanda LOGOUT** : sfîrşitul unei sesiuni de lucru.

Sintaxa :

$LOGO[UT] ⟨cr⟩

În cazul în care sesiunea de lucru se desfăşoară cu fişier jurnal, se va constitui un punct de control pentru această bază de date.

Comanda provoacă eliberarea resurselor utilizate şi aduce terminalul în starea iniţială (înainte de LOGIN) iar dacă, pe aceeaşi bază, sînt şi alte console active, editează mesajul TRAVAIL TERMINE Pxxx, unde xxx este numărul afectat de LOGIN.

5) **Comanda NUMBER** : cere afişarea numărului de utilizatori simultani a bazei de date.

Sintaxa :

$NUMB[ER] ⟨cr⟩

Comanda este utilă în momentul în care dorim să introducem datele unui nou utilizator al bazei de date.

6) **Comanda CLOCK** : afişarea orei sistemului central.

Sintaxa :

$CLOC[K] ⟨cr⟩

Sistemul comunică ora sub forma : hh.mm.ss.cc unde :

– hh : ora ;
– mm : minutul ;
– ss : secunda ;
– cc : sutimea de secundă.

7) **Comanda MAIL** : obţinerea mesajului zilei.

Sintaxa :

$MAIL ⟨cr⟩

8) **Comanda MESSOP** : transmiterea unui mesaj la operatorul central.

Sintaxa :

$MESSOP – ⟨*mesaj*⟩ ⟨cr⟩

Mesajul este afişat pe consola operatorului central precedat de coordonatele utilizatorului (numărul terminalului şi eventual numărul liniei).

9) **Comanda PAUSE** : cerere de constituire a unui punct de control.

Sintaxa :

$PAUSE ⟨cr⟩

Comanda provoacă blocarea tastaturii consolei pînă la declanşarea efectivă a CKPT şi transmiterea mesajelor :

– CKPT DUE... – la consola operatorului central ;
– PAUSE EN FIN DE REQUETE – la toate posturile active ;
– CKPT EN COURS – la orice tentativă de LOGIN.

10) **Comanda EVAL** : activarea procedurilor de evaluare a performanţelor.

Sintaxa :

$EVAL ⟨cr⟩

După fiecare compilare sau execuţie se imprimă mesajele :

T : *xx*/*yy*/*zz*
D : *dd*/*pp*

– *xx* : timp UC consumat ;
– *yy* : timp aparent ;
– *zz* : timpi de aşteptare a intrărilor/ieşirilor (în afara celor datoraţi terminalului) ;
– *dd* : numărul de accesuri la disc ;
– *pp* : numărul de accesuri la paginare.

Efectul comenzii EVAL se anulează prin comanda :

$NOEV[AL] (NL)

11) **Comanda ACTI** : editarea conţinutului (la dispozitivul de ieşire standard al calculatorului) registrelor şi a zonelor de memorie sau activarea unor opţiuni de prelucrare.

Sintaxa :

$ACTI TB : ** ⟨cr⟩

Un nivel activat de un ACTI poate fi anulat cu comanda :

$DACT[I] TB : ** ⟨cr⟩

Generarea bazei de baze

Se realizează prin rularea, sub controlul SOCGBDB, unui fișier de comenzi, pus la dispoziție de firmă, prin care se introduce structura bazei de baze (descrisă cu LDD-SOCRATE) și un set de macroinstrucțiuni care permit administratorului bazei de baze să creeze și să întrețină datele tehnice atașate bazelor de date, aflate în gestiunea bazei de baze și utilizatorilor acestor baze.

Structura generală a JOB-ului de generare a bazei de baze este :

```
.   * RESTAURARE SGBD-SOCRATE
.   INIT DVT:AD,VS:AD1AD1
.   SYSRUN MAINT
% REST VOL
% OLDFLE DV:MT1,FN: 'SOCRATE'
% NEWVOL DVT:AD, VS:AD1AD1
% END
.   * DEFINIRE PROCEDURA GENERALA DE APEL A UNUI MODUL SOCRATE
.   LIST
.   XPROC PROCGEN,DEF
.   FETCH LN:BIBIMTSO GN:&GN:1&,VN:&VN:0&FN:&,MODUL:SOCBTCH&,
            &SUPB:DVT:AD,VS:AD1AD1&
.   ASSIGN S,&SUPS:DVT:AD,VS:AD1AD1&
.   ASSIGN Q,&SUPS:DVT:AD,VS:AD1AD1&
.   LABEL S,AM:OFL,FN:'& FANS: BAZA DE BAZE&'
.   LABEL Q,AM:OFL,FN:'&FNQ:MANEVRA SGBD&'
.   ENDPROC
.   *ALOCARE SPATIU PENTRU BAZA DE BAZE
.   ALLOC DVT:AD,VS:AD1AD1,AM:ANY,SZ:12,FN:'BAZA DE BAZE'
.   * ALOCARE SPATIU PENTRU FISIER DE LUCRU SOCRATE
.   ALLOC DVT:AD,VS:AD1AD1,AM:ANY:SZ:12,FN:'MANEVRA SGBD'
.   * GENERARE BAZA DE BAZE
.   XPROC PROGEN,MOD                    apelul procedurii generale
.   MOD &MODUL:SOCGBDB&                 apelul modulului SOCGBDB
.   ENDMOD
.   OPTION CS'CF:20,DS:4096'
.   RUN TIME:999,NL:500000,AD:0,0
$
<nr 1>  16   FORMATUL LINIILOR PENTRU DECLARAREA DIMENSIUNII IN
<nr 2>   4   PAGINI A SUBSPATIILOR SI A DIMENSIUNII SUBPAGINII ESTE :
<nr 3>  256  <Nrpag> <lgpag>
DEBUT
 ENTITE 100 BASE /* 1 REALIZARE/FIECARE BAZA DE DATE IN PARTE       */
  DEBUT
   NOM MOT 7 /* NUMELE BAZEI DE DATE */
   BEB DEBUT
    ORG-STRUC DE 0 A 1999999999 /* ADRESA DE ORIGINE A STRUCTURII
                                   INTERNE                          */
    ORG-FICH  DE 0 A 1999999999 /* ADRESA DE ORIGINE A SPATIULUI FICH */
    ORG-MAC   DE 0 A 1999999999 /* ADRESA DE ORIGINE A MACROINSTRUC-
                                   TIUNILOR                         */
   ENTITE 10 XBEB /* 1 REALIZARE/SUBSPAŢII AL BAZEI DE DATE          */
    DEBUT
     TYPE (10 4) (FICH DICO PROG STRU) /* TIP SUBSPATIU              */
     NBPAG    DE 0 A 9999999 /* NUMAR DE PAGINI ALOCATE              */
     TSPAG    DE 0 A 256     /* DIMENSIUNE S-PAG UN CUVINTE (L-1)   */
     LONGSP   DE 0 A 9999999 /* DIMENSIUNE SPAG IN OCTETI            */
```

```
            LGCHBIT     DE 0 A 9999999  /* IN-                                  */
            DEP         DE 0 A 9999999  /* FOR-                                 */
            SLL         DE 0 A 9999999
            SLLPP2      DE 0 A 9999999  /* MA-                                  */
            SRLPP2      DE 0 A 9999999  /* TII                                  */
            SLL32Mk     DE 0 A 9999999  /*                                      */
            PP2MASK     DE 0 A 512      /*                                      */
            KMASK       DE -1999999999 A -13 /* TEH-                            */
            ORGREL      DE 0 A 99999999 /*       NICE                           */
         NB-CYL         DE 0 A 9999999  /* NUMAR DE CILINDRI ALOCATI            */
         FIN
     FIN
   SUPORT DEBUT /* CARACTERISTICILE SUPORTULUI BAZEI DE DATE                   */
    TYPE ( 9 6 ) (MD17 MD25 MD50  MD100) /* TIPUL SUPORTULUI                   */
        ENTITE 20 VOLUME */ 1 REALIZARE/FIECARE VOLUM ALOCAT                   */
          DEBUT                                                                */
            NOM-PROPRIO MOT 14/* NUME PROPRIETAR (ON)                          */
            NB-CYL DE 0 A 2000 /* NUMAR DE CILINDRI ALOCATI                    */
          FIN
        FIN
        ENTITE 50 UTILISATEUR /* UTILIZATORII BAZEI DE DATE                    */
          DEBUT               /* 1 REALIZARE/UTILIZATOR                        */
            NOM MOT 15        /* NUMELE UTILIZATORULUI                         */
            NUMERO DE 0 A 9999 /* NUMAR DE CONT                                */
            MOT-DE-PASSE MOT 7 /* PAROLA UTILIZATORULUI                        */
      DROIT-UTILISATEUR /* DREPTURI DE ACCES */
        DEBUT
          ACCES-MACRO (15 12) /* SUR 1 DEMI-OCTET */ /* LA MACRO               */
                      (IMT COM  IMT+COMP MAC IMT+MAC
                      COMP+MAC IMT+COMP+MAC RESTREINT)
         ACCES-LANGAGE ( 15 11) /* 1 DEMI-OCTET */ /* LA LIMBAJ                */
                      AUTORISE MISE-A-JOUR
                      CREATION DEFINITION)
  PW-UTILISATEUR DE 1 A 16777215 /* 3 OCTETS */ /* LA OPERATII                 */
        FIN
          FIN
   FIN
  PW1 MOT 20 /* PAROLE   */
  PW2 MOT 20 /*          */
  PW3 MOT 20 /*          */
  PW4 MOT 20 /* SISTEM   */
```

*) Prin ⟨nr_1⟩, ⟨nr_2⟩ şi ⟨nr_3⟩ se declară numărul de pagini pe 1ko alocat sub-spaţiului bazei de bazei astfel :

— ⟨nr_1⟩, pentru sub-spaţiul „fişier" (FICH), cu sub-pagini de 16 cuvinte ;

— ⟨nr_2⟩, pentru sub-spaţiul „dicţionar" (DICO), cu sub-pagini de 4 cuvinte ;

— ⟨nr_3⟩, pentru sub-spaţiul „program" (PROG), cu sub-pagini de 256 cuvinte.

```
FORMAL CARTE-VOL        DEBUT
                        IDENT MOT (3)
                        BID1 MOT (1)
                        TYPE MOT (5)
                        BID2 MOT (1)
                        NOM-PROPRIO MOT 14
                        BID3 MOT (1)
                        NB-CYL DILATE (3)
                        BID4 MOT (52)
                        FIN
```

Acest formal logic descrie structura cartelelor de tip VOL utilizate pentru declararea caracteristicilor de identificare şi tratare a suporţilor bazei de date ;

```
FORMAL CARTE-UTI DEBUT IDENT MOT (3)
                       BID1 MOT (1)
                       NOM MOT (15)
                       BID2 MOT (1)
                       NUMERO DILATE (4)
                       BID3 MOT (1)
                       MOT-DE-PASSE MOT (7)
                       BID4 MOT (1)
                       ACCES-MACRO MOT (17)
                       BID5 MOT (1)
                       ACCES-LANGAGE MOT (11)
                       BID6 MOT (1)
                       PW-UTILISATEUR DILATE (8)
                       BID7 MOT (14)
                       FIN
```

Acest formal logic descrie structura cartelelor de tip UTI utilizate pentru introducerea datelor de identificare ; autentificare și acordare a drepturilor de acces ai utilizatorilor bazei de date ;

```
FORMAL CARTE-ORG       DEBUT
                       IDENT MOT 3
                       BID1 MOT 3
                       ORG-STRUC DILATE 10
                       BID2 MOT 1
                       ORG-FICH DILATE 10
                       BID3 MOT 1
                       ORG-MAC DILATE 10
                       BID4 MOT 44
                       FIN
FIN
?
```

Acest formal logic descrie structura cartelelor de tip ORG care permit modificarea adreselor de origine ale sub-spațiilor bazei de date ;

Să presupunem că spațiul alocat pentru „BAZA DE BAZE" este divizat astfel :
FICH : 7 cilindri ;
DICO : 1 cilindru ;
PROG : 4 cilindri.

Valorile lui $\langle nr_1 \rangle$, $\langle nr_2 \rangle$, $\langle nr_3 \rangle$ reprezintă cel mai mare număr prim de pagini de 1 ko care se încadrează în numărul de cilindri rezervați corespunzător pentru spațiul respectiv.

Pentru exemplul dat avem :

– FICH : 7 cilindri ($7 * 120 = 840$ pagini de 1 ko) deci
  $\langle nr_1 \rangle$: 839 pagini cu sub-pagină de 16 cuvinte ;
– DICO : 1 cilindru ($1 * 120 = 120$ pagini de 1 ko) deci
  $\langle nr_2 \rangle$: 113 pagini cu sub-pagina de 4 cuvinte ;
– PROG : 4 cilindri ($4 * 120 = 480$ pagini de 1 ko) deci
  $\langle nr_3 \rangle$: 479 pagini cu sub-pagina de 256 cuvinte.

Pentru validarea acestor valori sistemul poartă la consolă, un dialog cu operatorul astfel :

nn SOCRATE A VOTRE SERVICE
– LO SOCRATE (NL)
nn TYPE DE DISQUE (MD17,MD25,MD50,MD100) ? MD50 (NL)
nn FICH NBR PAGES: $\langle nr_1 \rangle$ TAILLE SS PAGES 16 ? OK (NL) –
nn DICO NBR PAGES: $\langle nr_2 \rangle$ TAILLE SS PAGES 4 ? OK (NL)
nn PROG NBR PAGES: $\langle nr_3 \rangle$ TAILLE SS PAGES 256 ? OK (NL)
nn FIN FORMAT SISTEM OK
nn FIN DEF+QEST
nn CREATION BTCH ? NON (NL)

unde :

nn – **numărul partiției în care se lansează ;**

LO SOCRATE – este obligatoriu și semnifică realizarea conectării (LOGIN) cu sistemul SOCRATE ;

(NL) — tasta NEW-LINE ;
TYPE DE DISQUE — tipul discului pe care se face generarea bazei de baze (tipul va da dimensiunea paginii din spaţiul real — numărul de sectoare disc pe o pagină) ; valoarea furnizată trebuie să aparţină listei ;
FICH — numărul de pagini ale spaţiului fişier ;
DICO — numărul de pagini ale spaţiului dicţionar ;
PROG — numărul de pagini ale spaţiului program ;
CREATION BTCH ? — este impus răspunsul NON.

Orice alt răspuns decît OK respectiv NON provoacă abandonarea prelucrării iar utilizatorul trebuie să reia integral prelucrarea.

Introducerea parolelor de acces sistem

Pentru a se asigura discreţia datelor conţinute în baza de baze şi a bazelor de date aflate în gestiune, administratorul bazei de baze are la dispoziţie 4 parole sistem (PW1–PW4) prin care permite accesul total sau parţial la datele din baza de baze şi la anumite funcţiuni ale unor module (de exemplu, formatarea care înseamnă iniţializarea cu 0 — nu este permisă decît dacă se cunoaşte valoarea lui PW3).

Introducerea acestor parole se face apelînd sub controlul SGBD-SOCRATE (LO SOCRATE sau % SOC BN : SOCRATE) o macroinstrucţiune sistem al cărui apel în „batch processing" se face astfel :

```
. XPROC PROCGEN,MOD
.   MOD& MODUL:SOCBTCH&
.   ENDMOD
.   OPTION CS'CF:10,DS:1024'
.   RUN TIME:999,NL:500,AD:0,0
%   SOC BN:SOCRATE
%   RUN FN:O
  MODIF-PW SOCRATE ?
CURS
SOCRATE     PAROLE INTRODUSE CITE UNA PE LINIE
ABSOLVIT LA     (maxim 20 caractere/parola)
ASE-BUCURESTI
%   LOGOUT
```

Macroinstrucţiuni sistem pentru acces la datele conţinute în baza de baze

Orice acţiune asupra datelor conţinute în ⟨BAZA DE BAZE⟩ se face sub controlul direct al SGBD-SOCRATE (prin LOGIN SOCRATE, în conversaţional, sau %SOC BN : SOCRATE, în „batch processing") cu ajutorul unor macroinstrucţiuni ale sistemului puse la dispoziţia utilizatorilor.

Structura generală a unui macro sistem este :

⟨macro-sistem⟩ : : = *⟨model de apel⟩ ⟨model de expansiune⟩*

⟨model de expansiune⟩ : : = este scris în limbajul de manipulare a datelor şi respectă regulile de formare a modelului de expansiune de macrogenerator ;

⟨model de apel⟩ : : = *⟨nume macro sistem⟩ ⟨lista de parametri⟩* {*⟨nume macro sistem⟩ ⟨lista de parametri⟩*} . . . . . ?

⟨lista de parametri⟩ : : = *⟨param 4⟩*/*⟨param 1⟩ ⟨param 2⟩ ⟨param 3⟩ ⟨param 4⟩* /*⟨param 5⟩ ⟨param 1⟩ ⟨param 2⟩ ⟨param 3⟩ ⟨param 4⟩*

În această listă de parametri, valoarea parametrului ⟨param 4⟩ este reprezentată de valoarea uneia din parolele sistem (PW_i, $i = 1, 2, 3, 4$). Pentru utilizarea parolelor sistem, ca valori ale parametrilor, se face precizarea că utilizatorii care cunosc valoarea PW_i, a unei parole, au acces la orice macro care are ca parametru valoarea parolei PW_j dacă și numai dacă $i > j$. Administratorul bazei de baze care cunoaște valoarea lui PW4 are acces la toate macroinstrucțiunile sistemului. Dacă valoarea parolei cerute este a unei parole mai puternice, atunci sistemul editează mesajul:

MOT DE PASSE INCORRECT REQUETE INTERDITE...

iar acțiunea macro apelate este inefectivă.

Dacă valoarea unuia din parametrii ⟨*param i*⟩ ($i \neq 4$) este eronată, atunci se editează mesajul:

BASE UTILISATEUR MOT-DE-PASSE INCONNU... și acțiunea macro apelate este inefectivă.

Modificarea parolelor sistem

Poate fi realizată în orice moment apelînd sub LOGIN SOCRATE macroinstrucțiunea MODIF-PW avînd ca parametru parola sistem de cel mai înalt nivel (⟨PW4⟩). Este indicat ca această macroinstrucțiune să nu fie catalogată într-o procedură deoarece valorile atribuite parolelor sistem pot fi alterate datorită modului de tratare a șirurilor de caractere la obținerea expansiunii procedurii (transferul în fișierul SYSFTxx poate crea caractere parazite).

```
.    XPROC PROCGEN,MOD
.    MOD &MODUL:SOCBTCH&
.    ENDMOD
.    OPTION CS'CF:10 DS:1024'
.    RUN TIME:999,LN:500,AD:0,0
$
%    SOC BN:SOCRATE        (LOGIN SOCRATE)
MODIF-PW  PW4 ?
 valoare nouă PW 1
 valoare nouă PW 2
 valoare nouă PW 3
 valoare nouă PW 4
%    LOGOUT
```

Editarea datelor tehnice atașate bazelor de date gestionate de o bază de date și a parolelor sistem

FICHIER BDB ⟨*param* 4⟩ [⟨*macro sistem*⟩] ?

⟨*param* 4⟩ : : = valoarea parolei sistem PW4

Macroinstrucțiunea editează (în sens ⟨*entitate*⟩ SOCRATE):

– caracteristicile tehnice atașate bazelor de date gestionate pentru fiecare bază de date în parte:

- numele bazei de date;
- caracteristicile spațiilor bazei de date (dimensiune subpagină, număr de pagini rezervate, număr de cilindri alocați etc.);
- datele caracteristice ale suportului (suporților) bazei (tipul suportului, numele proprietarului, numărul de cilindri alocați bazei de date);

- datele asociate utilizatorilor bazei de date (numele utilizatorului, numărul de cont, parola, drepturile de acces la baza de date) ;
- valorile atașate parolelor sistem (PW1, PW2, PW3 și PW4).

Exemplu :

```
% RUN FN:R
FICHIER BDB ASE-BUCURESTI ?
```

Editarea datelor tehnice atașate unei baze de date

EDIBASE ⟨*param* 1⟩ ⟨*param* 2⟩ ⟨*param* 3⟩ ⟨*param* 4⟩ [?]
Editează (în sens ⟨*entitate*⟩) caracteristicile tehnice ale bazei de date citate în valoarea ⟨*param* 1⟩.
⟨*param* 1⟩ : : = numele bazei de date căreia dorim să-i edităm caracteristicile tehnice ;
⟨*param* 2⟩ : : = numele utilizatorului ;
⟨*param* 3⟩ : : = parola utilizatorului ;
⟨*param* 4⟩ : : = cel puțin valoarea parolei sistem PW1.

Exemplu :

```
EDIBASE BDDPERS ASE ASE CURS ?
```

Adăugarea în conversațional (sau simulare „batch processing") a unui utilizator al unei baze de date

CREATION-UTIL DE LA BASE ⟨*param* 1⟩ ⟨*param* 2⟩ ⟨*param* 3⟩ ⟨*param* 4⟩ [⟨*macro-sistem*⟩] ?
⟨*param* 1⟩ : : = numele bazei de date pentru care se adaugă un utilizator ;
⟨*param* 2⟩ : : = numele utilizatorului care crează ;
⟨*param* 3⟩ : : = parola sa ;
⟨*param* 4⟩ : : = cel puțin valoarea parolei sistem ⟨PW 2⟩.

Crează (în sensul funcției de ⟨CREARE⟩ a SGBD-SOCRATE) o realizare a entității ⟨UTILISATEUR⟩, din structura bazei de baze, și cere prin dialog la consolă (sau prin simulare a dialogului, pe linii tip intrare standard în „batch processing") valorile caracteristicilor declarate în această entitate (sînt cerute și răspunsurile la întrebările procesului de creare iar pentru valorile furnizate de utilizator efectuează și validarea în sens SOCRATE).

Exemplu :

```
QUESTION:CREATION-UTIL DE LA BASE BDDPERS AVRAM SOC ASE-BUCURESTI ?
        CREATION UN UTILISATEUR ? OUI ⟨cr⟩
          NOM :COJOCARU ⟨cr⟩
          NUMERO:21 ⟨cr⟩
          MOT-DE-PASSE:#1#XY3Z ⟨cr⟩ (parola)
          CREATION DROIT-UTILISATEUR ? OUI ⟨cr⟩
            ACCES-MACRO : IMT+COMP+MAC ⟨cr⟩
            ACCES-LANGAGE : DEFINITION ⟨cr⟩
            PW-UTILISATEUR : 1013377 ⟨cr⟩
            (parola de acces la limbaj).
```

Adăugarea în „batch processing" a unui utilizator al bazei de date

CREABTCH-UTIL DE LA BASE ⟨*param* 1⟩ ⟨*param* 2⟩ ⟨*param* 3⟩ ⟨*param* 4⟩ [⟨*macro-sistem*⟩] ?
⟨*param* 1⟩ : : = numele bazei de date ;
⟨*param* 2⟩ : : = numele utilizatorului care crează ;
⟨*param* 3⟩ : : = parola utilizatorului care crează ;
⟨*param* 4⟩ : : = cel puţin valoarea lui ⟨PW2⟩.

Citeşte linii de tip UTI şi crează, pentru fiecare din ele, un nou utilizator al bazei de date, pînă la întîlnirea liniei de tip EOF.

Ştergerea unui utilizator al unei baze de date

SUPUTIL ⟨*param* 5⟩ DE LA BASE ⟨*param* 1⟩ ⟨*param* 2⟩ ⟨*param* 3⟩ ⟨*param* 4⟩ [⟨*macro-sistem*⟩] ?
⟨*param* 1⟩ : : = numele bazei de date ;
⟨*param* 2⟩ : : = numele utilizatorului care efectuează ştergerea ;
⟨*param* 3⟩ : : = parola utilizatorului care efectuează ştergerea ;
⟨*param* 4⟩ : : = cel puţin valoarea parolei sistem PW3 ;
⟨*param* 5⟩ : : = numele utilizatorului care este şters.

Realizează ştergerea (suprimare în sensul LMD) primului utilizator care are numele ⟨*param* 5⟩ şi are acces la baza de date ⟨*param* 1⟩.

Exemplu :

SUPUTIL ASE DE LA BASE ADMITBD
SABAU SOC ASE-BUCURESTI
FICHIER BDB ASE-BUCURESTI ?

Modificarea datelor ataşate unui utilizator

MODIF-UTIL ⟨*param* 5⟩ DE LA BASE ⟨*param* 1⟩ ⟨*param* 2⟩ ⟨*param* 3⟩ ⟨*param* 4⟩ [⟨*macro-sistem*⟩] ?
⟨ *param* 1⟩ : : = numele bazei de date ;
⟨*param* 2⟩ : : = numele utilizatorului care efectuează modificarea ;
⟨*param* 3⟩ : : = parola celui care efectuează modificarea ;
⟨*param* 4⟩ : : = cel puţin valoarea parolei sistem PW3 ;
⟨*param* 5⟩ : : = numele utilizatorului căruia i se modifică datele.

Realizează modificarea prin dialog la terminal sau simulare „batch processing", a datelor aferente primului utilizator al bazei de date ⟨*param* 1⟩ cu numele ⟨*param* 5⟩.

Exemplu :

QUESTION:MODIF-UTIL SABAU DE LA BASE ADMITBD COJOCARU ⟨cr⟩
#1#XY3Z ASE-BUCURESTI ? ⟨cr⟩

Scoaterea unei baze de date din evidenţa unei baze de baze

SUPBASE ⟨*param* 1⟩ ⟨*param* 2⟩ ⟨*param* 3⟩ ⟨*param* 4⟩ [⟨*macro sistem*⟩] ?
⟨*param* 1⟩ : : = numele bazei de date care se şterge ;
⟨*param* 2⟩ : : = numele utilizatorului care efectuează operaţia.

⟨*param* 3⟩ : : = parola utilizatorului ;
⟨*param* 4⟩ : : = cel puțin valoarea parolei sistem PW3.

Baza de date cu numele ⟨*param* 1⟩ va fi scoasă din gestiunea bazei de date (ștergerea în baza de baze este făcută în sensul suprimării).

Utilitare care funcționează sub controlul modulului XXXGBAS

Acest modul pune la dispoziția utilizatorului următoarele funcțiuni :

1) generarea bazei de date ;

2) formatarea parțială sau totală a bazei de date/subspațiilor ;

3) statistici parțiale sau totale, pe o bază de date și determinarea coerenței fizice ;

4) dump parțial sau total al conținutului paginilor unei baze de date/subspațiu ;

5) reorganizarea parțială sau totală a bazei de date prin intermediul operațiilor de salvare-restaurare.

Fiecare funcțiune are asociat un procesor care poate fi adaptat necesităților de execuție ale utilizatorului prin poziționarea unor parametrii de execuție. Lansarea în execuție a unui procesor se realizează cu ajutorul comenzii % SYSRUN.

Comanda SYSRUN

Rol : lansarea în execuție a procesoarelor modulului XXXGBAS.
Sintaxa :
% [⟨*eticheta*⟩] SYSRUN FN : ⟨*procesor*⟩ [,ST : ⟨*sub-spațiu*⟩ [,BS : (*n*, *m*)]]
⟨*procesor*⟩ : : = F/S/D/V/R
F : formatare ;
S : statistică (coerență fizică) ;
D : dump ;
V : salvare ;
R : restaurare.
⟨*sub-spațiu*⟩ : : = FICH/DICO/PROG/ALL
FICH : subspațiul „fișier" ;
DICO : subspațiul dicționar ;
PROG : subspațiul program ;
ALL : toate sub-spațiile bazei de date.

Dacă argumentul ST nu este specificat atunci acțiunea funcțiunii solicitate se va desfășura asupra întregului spațiu alocat bazei de date, tratînd specific fiecare subspațiu al acesteia.

BS : precizează care subpagini, din sub-spațiul specificat, reprezintă obiectul acțiunii comenzii.

Acțiunea se va desfășura asupra tuturor paginilor din intervalul [*n*, *m*] cu $n \leqslant m$. Dacă $n = m$ atunci va fi tratată o singură pagină. Modul de acțiune al procesoarelor va fi prezentat, pentru fiecare în parte, în paragrafele următoare.

Generarea unei baze de date

Realizarea acestei operaţii este sarcina administratorului bazei de baze (cunoscător al parolei PW4) care primeşte, de la administratorul bazei de date, următoarele informaţii necesare la momentul apelului modulului de generare :

— numărul de cilindri alocaţi bazei de date (eventual chiar comanda ALLOC) ;

— numele afectat bazei de date ;

— comenzile VOL şi UTI, EOF şi % ALLOC.

După generarea şi formatarea noii baze, responsabilul bazei de baze furnizează responsabilului bazei de date valorile parolelor sistem ⟨PW1⟩, ⟨PW2⟩ şi eventual ⟨PW4⟩ pentru a-i permite acestuia să aibă accesul la macroinstrucţiunile de modificare a conţinutului baze de baze pentru a introduce noi utilizatori ai bazei de date, a modifica datele afectate celor definiţi deja etc.

Din punct de vedere al SGF, o bază de date este un fişier nedefinit pentru care mărimea înregistrării fizice şi logice este de 1 024 octeţi. Din punct de vedere al sistemului SOCRATE, spaţiul alocat unei baze de date este divizat logic şi fizic (prin datele tehnice asociate bazei de date în baza de baze) în trei subspaţii FICH (fişier), DICO (dicţionar) şi PROG (program). Dimensiunea fizică în unităţi elementare (subpagini) logice, este furnizată pentru fiecare subspaţiu în comanda % ALLOC. În funcţie de numărul de cilindri şi de dimensiunea aleasă pentru subpagină, sistemul calculează în mod automat numărul de subpagini disponibile în fiecare spaţiu în parte. Dimensiunea subpaginii este prefixată în cazul subspaţiilor DICO (16 octeţi) şi PROG (256 octeţi) şi la dispoziţia opţiunii utilizatorului în cazul subspaţiului FICH. Modul de utilizare a subspaţiilor (dat de datele pe care le gestionează) este :

— FICH : codul intern al structurii bazei de date, codul sursă al macro-instrucţiunilor şi programelor precompilate definite de utilizator, datele asociate structurii bazei de date (informaţiile stocate de utilizator) ;

— DICO : dicţionare de acces pentru caracteristicile declarate în structură chei de acces, dicţionare de acces pentru macroinstrucţiuni, programe precompilate, programe IMT ;

— PROG : programe precompilate şi programe IMT.

Comanda VOL

Permite introducerea datelor de identificare a suportului/suporţilor bazei de date. Dacă baza de date ocupă mai multe volume atunci fiecărui volum i se va asocia comanda VOL corespunzătoare.

Structura şi semnificaţia valorilor furnizate cu ajutorul comenzii VOL este redată în tabela 6.2.

Tabela **6.2. Parametrii comenzii VOL**

Coloanele	Semnificaţie
1	2
1— 3	VOL (identificator comanda)
4	spaţiu
5— 9	tipul discului : MD17, MD25, MD50, MD100

Tabelul **6.2** *(continuare)*

1	2
10	spaţiu
11–24	numele proprietarului volumului (argumentul ON din comanda INIT)
25	spaţiu
26–28	Numărul de cilindri, alocaţi bazei pe suportul respectiv (maxim 3 cifre zecimale, aliniat la dreapta si completat eventual cu zerouri nesemnificative)
29–80	spaţiu

Comanda UTI

Pentru a proteja informaţiile conţinute într-o bază de date, sistemul SOCRATE pune la dispoziţia administratorului bazei de date următoarele utilităţi :

— descrierea drepturilor de acces ataşate utilizatorilor bazei de date ;

— posibilitatea testării drepturilor de acces la o cerere de consultare a informaţiilor bazei de date sau la apelul macro-instrucţiunilor şi programelor precompilate.

Introducerea utilizatorilor şi a drepturilor lor de acces se realizează prin apelarea funcţiunilor corespunzătoare (la generarea bazei de date sau prin apelul macro de acces la informaţiile din baza de baze) ale sistemului care tratează comenzile de tip UTI. Structura şi semnificaţia cîmpurilor comenzii UTI este redată în tabelul 6.3.

Tabelul **6.3. Semnificaţia cîmpurilor comenzii UTI**

Coloane	Semnificaţie
1–3	UTI (identificator comanda)
4	Spaţiu
5–19	Nume utilizator
20	spaţiu
21–24	cont utilizator
25	spaţiu
26–32	parola
33	spaţiu
34–45	acces macro
46	spaţiu
47–57	acces limbaj
58	spaţiu
59–66	PW — utilizator (valoare zecimală : $1 - (2^{24} - 1)$
67–80	spaţiu

Acces macro (1 semi-octet) : în funcţie de valoarea parametrilor furnizaţi în coloanele 34–45 ale comenzii UTI, sistemul completează semioctetul ACCES-MACRO, din structura bazei de baze, cu o valoare binară, conform tabelului 6.4.

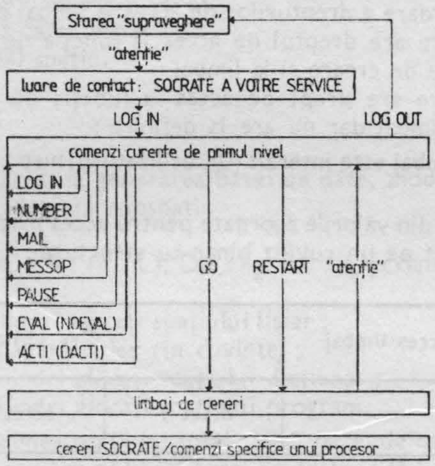


Fig. 6.2. Utilizarea comenzilor SOCRATE în conversaţional

Tabelul **6.4. Valoarea binară a semioctetului ACCES-MACRO**

Valoare binară	Valoare argument	Semnificaţie
0	spaţiu	apel macrogenerator interzis
8	RESTREINT	acces la ordinele de listare şi editare (LISMAC, LISPRO, LISMIT, LISALL, EDIMAC, EDIPRO)
1	IMT	acces la ordinele RESTREINT şi DEFIMT, SUPIMT, SUPEXP pentru programe IMT
2	COMP	acces la ordinele RESTREINT şi DEFPRO, SUPPRO, SUPEXP pentru programe precompilate
3	IMT + COMP	acces la ordinele IMT şi COMP
4	MAC	acces la ordinele RESTREINT şi DEFMAC, SUPMAC, SUPEXP pentru macro
5	IMT + MAC	acces la ordinele IMT şi MAC
6	COMP + MAC	acces la ordinele COMP şi MAC
7	IMT + COMP + MAC	nici o restricţie

Acces limbaj (1 semi-octet): în funcţie de valoarea parametrului furnizat în coloanele 47—57 ale comenzii UTI, sistemul completează semioctetul conform tabelei 6.5.

Tabelul **6.5. Valoarea binară şi semnificaţia acesteia pentru drepturile de acces**

Valoare binară	Valoare argument	Semnificaţie
0	spaţiu	nici un drept
1	AUTORISE	drept de acces la limbajul de căutare
2	MIS-A-JOUR	drept de acces la actualizare
3	CREATION	drept de acces la funcţia de creare
4	DEFINITION	drept de acces la funcţia de definire a structurii

Convenția de acordare a drepturilor de acces la limbaj este următoarea:

— utilizatorul care are dreptul de acces la funcția de definire, are implicit drept de acces la funcția de creare și la limbaj;

— utilizatorul care are drept de acces la funcția de creare (CREATION), are drept de acces la limbaj dar nu are la definire;

— dacă accesul limbaj este interzis (0), utilizatorul nu poate utiliza decît macroinstrucțiuni.

Ansamblul format din valorile acordate pentru acces macro, acces limbaj și PW utilizator, este memorat pe un cuvînt binar cu structura:

<table>
<tr><td>acces macro</td><td colspan="2">acces limbaj</td><td colspan="2">PW utilizator</td></tr>
<tr><td colspan="2">octet 0</td><td>octet 1</td><td>octet 2</td><td>octet 3</td></tr>
</table>

Sistemul SOCRATE efectuează, în mod automat control pe valorile acces macro și acces limbaj. Responsabilul bazei de date poate verifica cu ajutorul unui program IMT valoarea elementului PW utilizator (ultimii 3 octeți) care constituie elementul de asigurare a discreției asupra datelor din bază.

Comanda ORG

Permite modificarea adreselor de origine implicite ale elementelor din sub-spațiul FICH. Structura și semnificația cîmpurilor comenzii ORG este redată în tabela 6.6.

Tabelul **6.6. Structura și semnificația cîmpurilor comenzii ORG**

COLOANE	SEMNIFICAȚIE
1—3	ORG (identificatorul machetei)
4	spațiu
5—14	adresa de origine a spațiului alocat stocării structurii interne (*as*)
15	spațiu
16—25	adresa de origine a spațiului alocat stocării datelor (*ad*)
26	spațiu
27—36	adresa de origine a spațiului alocat stocării surselor macro și programelor precompilate (*am*)
37—80	spațiu

Relația de ordine care trebuie să existe între aceste valori este: $as < ad < am$.

Comanda EOF

Comenzile ORG, VOL și/sau UTI trebuie să se termine cu o comandă EOF, care anunță terminarea operației de introducere a datelor despre volumele și/sau utilizatorii bazei.

Structura liniei de comandă este :

Coloanele : 1–3 caracterele EOF
4–80 spaţiu.

Comanda ALLOC

Permite specificarea, la generarea bazei de date, modului de împărţire a spaţiului alocat bazei de date în subspaţii.

Sintaxa :

% [⟨*eticheta*⟩] ALLOC FA : (n_1, L), DA : n_2, PA : n_3 [⟨*comentariu*⟩]

Unde :

– n_1 : numărul de cilindri alocaţi spaţiului fişier ;
– L : mărimea sub-paginii fişier (în cuvinte) ;
– n_2 : numărul de cilindri alocaţi spaţiului dicţionar ;
– n_3 : numărul de cilindri alocaţi spaţiului program.

Dacă sz este mărimea spaţiului total alocat bazei de date (valoarea parametrului SZ : din comanda monitor •ALLOC), atunci trebuie să fie îndeplinită următoarea condiţie :

$$(n_1 + n_2 + n_3) \leqslant sz.$$

Generarea bazei de date BDDPERS

Să presupunem că trebuie să generăm baza de date BDDPERS de 50 cilindri. Modul de împărţire a acestui spaţiu în subspaţii (FICH, DICO, PROG) este specificat în comanda ALLOC iar generarea bazei se va face pe discul AD1. Liniile de comandă necesare generării sînt :

```
.     ALLOC DV:AD1,FN:'BDDPERS' ,AM:ANY,SZ:50
.     XPROC PROCGEN,MOD ⟨APEL SOCGBAS⟩
.     MOD &MODUL:SOCGBAS&
.     ENDMOD
.     OPTION CS'CF:20,DS:4096'
.     RUN TIME:999,NL:500000,AD:0,0
%     SOC BN:BDDPERS
VOL     MD50     050
UTI  SABAU   0010 SGBD  IMT+COMP+MAC   DEFINITION  00000099
UTI  AVRAM   0001 SOC   IMT+COMP+MAC   DEFINITION  00000001
UTI  ASE     1555 ASE   RESTREINT      DEFINITION      1555
EOF
%     ALLOC FA:(30,16),DA:10,PA:10
%     SYSRUN FN:F
%     LOGOUT
```

La întîlnirea comenzii % SOC, care anunţă o cerere de conectare la o bază de date, SGBD-SOCRATE verifică dacă această bază există în gestiunea bazei de baze. Dacă baza nu există, atunci sistemul generează o realizare a entităţii BASE, din baza de baze, căreia îi completează datele tehnice asociate în conformitate cu valorile argumentelor furnizate pe liniile de comandă VOL, UTI, ORG şi % ALLOC. Generarea bazei de date este validată numai la o furnizare corectă a parolei sistem PW4 cerută la consola sistemului prin mesajul MOT DE PASSE, (valoarea lui PW4 se dă în continuarea mesajului – fără spaţiu). După generarea bazei de date, este necesară efectuarea unei formatări a acesteia, formatare care reprezintă operaţia de constituire a

unor liste libere, corespunzătoare subspațiilor bazei de date, inițializate la valoarea 0 binar (nedefinit). Pentru fiecare pagină formatată se rezervă, la începutul său, un număr de cuvinte egal cu numărul de subpagini din pagină reprezentînd cuvintele de control. Rezervarea acestor cuvinte permite pe de o parte gestiunea înlănțuirii sub-paginilor iar pe de altă parte recuperarea automată a lor la ștergerea conținutului (sistem cu autoorganizare).

Formatare parțială sau totală a bazei de date

Am arătat în paragraful precedent în ce constă operația de formatare. Declanșarea acestei operații este necesară la :

— definirea inițială a bazei de date ;

— definirea unei structuri diferite, față de cea existentă în baza de date, care nu mai concordă cu datele reale stocate (modificare a lungimii diverselor elemente din structură) ;

— renunțarea la informațiile stocate în unul din subspațiile bazei de date sau într-o porțiune a unui subspațiu.

De exemplu, structura liniilor de comandă necesare pentru formatarea bazei de date BDDPERS este următoarea :

```
.  XPROC PROCGEN,MOD
.  MOD &MODUL:SOCGBAS&
.  ENDMOD
.  OPTION CS'CF:10 DS:4096'
.  RUN TIME:999,NL:500000,AD:0,0
%  SOC BN:BDDPERS,AN:1,PW:SOC,PN:AVRAM
%  SYSRUN FN:F [,ST: ⟨subspațiu⟩][,BS:(n,m)]
%  LOGOUT
```

⟨*subspațiu*⟩ : : = FICH/DICO/PROG

În cazul în care argumentul ST este omis acțiunea procesorului de formatare se desfășoară asupra întregului spațiu real al bazei de date ; în caz contrar (argumentul ST prezent) acțiunea procesorului se desfășoară numai pe subspațiul specificat.

Dacă este prezent argumentul BS (numai în combinație cu ST) atunci acțiunea se va desfășura numai asupra paginilor din intervalul $[n, m] \cap N$ din subspațiul respectiv.

Statistici parțiale sau totale

Utilitarul își desfășoară acțiunea asupra unui subspațiu, asupra întregului spațiu alocat al bazei de date sau asupra unei fracțiuni și furnizează informații tehnice privind (anexa 6) :

— numărul de subpagini ocupate ;

— numărul de subpagini de tip squatter ;

— numărul de subpagini al căror început este vid (din cele ocupate) ;

— numărul de subpagini al căror sfîrșit este vid.

Pentru aceste informații, furnizează și procentul pe care îl reprezintă față de total. Sistemul semnalează lanțul cu cele mai multe elemente iar pentru lanțuri furnizează, pe număr de înlănțuiri, numărul lor, numărul de schimbări de pagină și numărul de schimbări pe cilindru care trebuiesc efectuate în vederea parcurgerii lanțului.

Utilizatorul furnizează datele necesare :

— analizei coerenţei fizice a bazei de date ;

— analizei oportunităţii construirii unor noi metode de acces (definirea unor noi căi de acces) ;

— analizei ocupării subspaţiilor bazei de date.

Structura generală a liniilor de comandă necesare lansării procesorului de statistici pentru baza de date ADMITBD, de exemplu, este :

```
.  NLIST
.  XPROC PROCGEN,MOD
.  MOD &MODUL:SOCGBAS&
.  ENDMOD
.  OPTION CS'CF:10,DS:4096'
%  SOC BN:ADMITBD,AN:2499,PN:AVRAM,PW:SOC
%  SYSRUN FN:S [,ST: ⟨subspaţiu⟩] [,BS: n,m)]
%  LOGOUT
```

— FN:S — apel procesor statistică
— ST — subspaţiul analizat
— BS:(*n*,*m*) — *m* ⩾ *n*, precizează pentru spaţiul/subspaţiul, numărul de pagini care constituie fracţiunea pe care se desfăşoară acţiunea programului. Analiza procesorului se desfăşoară inclusiv asupra paginii *m* (numerotarea paginilor începe de la 0).

Rezultatul acţiunii acestui procesor, în cazul analizei tuturor subpaginilor bazei de date ADMITBD se prezintă ca în anexa 6.

Dump parţial sau total

Utilitarul furnizează pentru fiecare pagină, din spaţiul acţiunii, numărul său iar pentru subpaginile conţinute de pagină (numai pentru cele ocupate) editează :

— numărul subpaginii ;

— adresa virtuală asociată ;

— numărul paginii următoare ;

— numărul subpaginii următoare ;

— dacă subpagina este squatter sau nu ;

— conţinutul subpaginii în hexa şi clar.

Dacă subpagina nu este „squatter" şi este singura din lanţ, valoarea înregistrată în cîmpul ⟨*numărul paginii*⟩ este numărul paginii curente iar numărul subpaginii următoare este chiar numărul subpaginii.

Structura liniilor de comandă necesare lansării procesorului de efectuare a dump-ului este :

```
.  XPROC PROCGEN,MOD
.  MOD &MODUL:SOCGBASE&
.  ENDMOD
.  OPTION CS'CF:10,DS:4096'
.  RUN TIME:999,NL:500000,AD:0,0
%  SOC BN:BDDPERS,AN:1,PN:AVRAM,PW:SOC
%  SYSRUN FN:D [,ST: ⟨subspaţiu⟩] [,BS: n,m)]
%  LOGOUT
```

— FN:D — apel procesor realizare dump ;
— celelalte argumente au semnificaţie dată în paragraful anterior.

În anexa .7 este prezentat un exemplu de dump obţinut în mod automat în urma detectării unor incoerenţe fizice.

Reorganizarea bazei de date

Această acțiune este realizată, de obicei, foarte rar și poate fi declanșată de :

— diminuarea performanțelor de răspuns a bazei de date ;

— luarea deciziei de modificare a dimensiunii subpaginii pentru un subspațiu ;

— luarea deciziei de modificare a dimensiunii fizice a spațiului real sau a unui subspațiu al bazei de date ;

— existența unor inconsistențe a căror refacere n-a fost realizată, din diverse motive, la momentul apariției lor, cu ajutorul fișierelor jurnal.

Indiferent care este motivul pentru care se decide reorganizarea acțiunile efectuate de administrator sînt următoarele :

— se salvează datele conținute de baza de date, subspațiu cu subspațiu, cu ajutorul utilitarului de salvare ;

— se efectuează eventual, modificările dorite în dicționarul datelor în sensul introducerii noilor caracteristici ale subspațiilor bazei de date ;

— se formatează subspațiile în cauză ;

— se restaurează datele salvate, subspațiu ca subspațiu, cu ajutorul utilitarului de restaurare.

O facilitate importantă oferită este aceea de a decupa subspațiul în zone disjuncte adiacente și a salva/restaura aceste zone în întreg subspațiul (utilizarea argumentului BS la salvare).

Ordinea de salvare/restaurare a acestor zone nu este importantă. Important este faptul de a nu preciza argumentul BS la restaurare, mod în care restaurarea ia în considerare și reorganizează datele în întreg spațiul receptor. Dacă argumentul BS este menținut la restaurare atunci este posibil ca rezultatul acțiunii să nu ofere performanțele așteptate.

Salvarea parțială sau totală a spațiilor bazei de date

Pentru exemplificare considerăm că dorim să salvăm datele bazei de date BDDPERS în vederea transferării lor în baza de date ADMITBD ale cărei caracteristici tehnice atașate diferă de cele atașate lui BDDPERS.

Salvarea datelor din subspațiile bazei de date BDDPERS se va efectua utilizînd o secvență de linii de comandă cu următoarea structură :

```
.   XPROC PROCGEN,MOD
.   MOD &MODUL:SOCGBAS&
.   ENDMOD
.   OPTION CS'CF:10,DS:2100'
.   RUN TIME:999,NL:500000,AD:0,0
%   SOC BN:BDDPERS,AN:1,PN:AVRAM,PW:SOC
%   ATTACH VBAS
%   FILE PRM:OUT,RCS:n,BFS:2048,RCF:FIX
%   ASSIGN DV:MTXX
%   LABEL FN: '<nume>'
%   SYSRUN FN:V,ST: <subspațiu> [, BS:(n,m)]
%   LOGOUT
```

— RCS : **dimensiunea în octeți a subpaginii din subspațiul considerat RCS pentru subspațiile din BDDPERS sînt respectiv :**
FICH : n = 64 (LONGSP = 60)
DICO : n = 16
PROG : n = 1024

— MT** : adresa logică a unităţii de bandă magnetică pe care se realizează salvarea subspaţiului considerat ;
<nume> : numele acordat fişierului receptor al vidajului subspaţiului ;
<subspaţiu>::= FICH/DICO/PROG
FN:V — apel procesor de vidare a conţinutului subspaţiului.

Deci, pentru o bază de date, vom avea minim trei apeluri ale procesorului şi vor rezulta minim trei fişiere corespunzătoare celor trei subspaţii ale bazei de date.

Numărul de apeluri şi fişiere obţinute creşte direct proporţional cu numărul diviziunilor în zone a subspaţiilor.

Observaţi.

Dimensiunea în octeţi a subpaginii unuia din subspaţiile tratate poate fi obţinută rulînd, pentru baza de date, macroinstrucţiunea sistem EDIBASE sau pentru toate bazele conţinute în baza de baze macroinstrucţiunea sistem FICHIER BDB.

Pentru fiecare subspaţiu valoarea lui RCS va fi luată egală cu cea a caracteristicii LONGSP asociate, plus 4.

Mai mult dacă utilizatorul doreşte să decupeze subspaţiul în zone disjuncte atunci pentru fiecare subspaţiu marginile zonelor trebuie să fie cuprinse în intervalul [0, <NBPAG>], unde <NBPAG> este valoarea asociată caracteristicii NBPAG, din baza de baze, minus 1.

Pentru exemplul nostru spaţiul FICH are dimensiunea de 839 pagini cu numere în intervalul [0, 838]. Acest interval dorim să-l divizăm în două zone :

— o primă zonă de 300 de pagini ;

— o a doua zonă cu 539 pagini.

În acest caz argumentul BS va fi BS : (0,299) şi respectiv BS : (300,838). În cazul salvării pe zone va fi editat sistematic mesajul de atenţionare "IL EXISTE DE SQUATTER NON CHAINE" care nu trebuie să fie luat în considerare.

Indiferent care este modul de realizare a procesului de salvare se va edita automat rezultatul efectuării statisticii subspaţiului/zonei.

Restaurare parţială sau totală a subspaţiilor bazei de date

Această acţiune are rolul de a restaura datele salvate cu ajutorul utilitarului de salvare în zona destinată recepţionării lor.

Pentru exemplul nostru considerăm că datele salvate se încarcă în baza de date ADMITBD, subspaţiu cu subspaţiu. Pentru baza de date ADMITBD vom considera următoarele lungimi (în octeţi) asociate dimensiunii subpaginii pe subspaţii, astfel:

FICH : $n = 32$

DICO : $n = 16$

PROG : $n = 1024$

Structura cartelelor de comandă necesare reconstituirii bazei de date este

```
.  XPROG PROCGEN,MOD
.  MOD &MODUL:SOCGBAS &
.  ENDMOD
.  OPTION CS'CF:10,DS:2100'
.  RUN TIME:999,LN:500000,AD:0,0
%  SOC BN:ADMITBD
%  ATTACH VBAS
%  FILE PRM:INP,BFS:2048,RCF:FIX,RCS:n
%  ASSIGN DV:MT**
%  LABEL FN: '<nume>'
%  SYSRUN FN:R,ST: <subspaţiu> [,BS:(n,m)]
%  LOGOUT
```

— RCS : *n* — dimensiunea în octeţi a subpaginii din subspaţiul tratat. Această dimensiune este cea a subpaginii din noua bază de date (poate fi diferită de cea pe care o avea la realizarea salvării numai pentru subspaţiul FICH).

— MT : unitatea logică de bandă magnetică pe care se află montat fişierul ⟨*nume*⟩ corespunzător subspaţiului din baza de date ⟨*subspaţiu*⟩.**

— FN:R — apel procesor de reconstituire (restaurare) a unui subspaţiu dintr-un vidaj efectuat anterior.

Observaţie

Deşi operaţia este de tip „restaurare" considerăm termenul „reconstituire" mult mai sugestiv în sensul că procesorul de restaurare execută plasarea „squatterilor" la locul lor, ori de cîte ori acest lucru este posibil, rezultînd astfel o reorganizare a spaţiului real.

Operaţiile de salvare/restaurare pe zone sînt indicate pentru orice operaţie de reorganizare a unor baze de date cu spaţiu real voluminos.

O dimensiune optimă a unei zone este de circa 10 000–15 000 de pagini pentru o durată de prelucrare în jur de o oră. În acest mod prelucrările voluminoase pot fi divizate în „mini sesiuni", reluarea în caz de incident fiind la nivel de „mini sesiune".

Durata de execuţie poate diferi de la o „mini sesiune" la alta în funcţie de procentul de ocupare al subpaginilor din zona prelucrată.

Posibilităţi de refacere a incoerenţelor fizice

Semnalarea existenţei incoerenţelor fizice precum şi a locului fizic al producerii lor (numărul paginii şi al subpaginii) este realizată de către utilitarul de statistici (anexa 7). Administratorul bazei de date analizează aceste incoerenţe iar în cazul în care consideră că nu sînt deosebit de importante sau nu mai are altă soluţie poate realiza o reorganizare totală sau parţială a subspaţiului considerat. Zona din subspaţiul şi mărimea sa este determinată de modul de plasare a numerelor paginilor eronate. Dacă reorganizarea se efectuează pe zone atunci operaţiile pe care trebuie să le execute sînt :

— se salvează cu ajutorul utilitarului de salvare datele din fiecare zonă ;
— se formatează cu ajutorul utilitarului de formatare fiecare zonă ;
— se restaurează cu ajutorul utilitarului de restaurare datele salvate.

Cu toate că este posibilă această tratare pe zone nu este indicată decît în cazuri extreme. Este preferabilă reorganizarea întregului subspaţiu care conţine incoerenţe fizice.

Indiferent care este metoda de refacere după realizarea operaţiilor cerute se va executa testul de coerenţă fizică, pe întreg subspaţiul asupra căruia s-a acţionat, cu ajutorul utilitarului de statistică.

Dacă rezultatul afişat de acesta este corect atunci este imperios necesară rularea bateriei complete de teste logice pentru subspaţiul reorganizat. Modul de construire a programelor „speciale" de testare a coerenţei logice este prezentat în capitolul 8.

Implementarea unei baze pe disc în V 1.6.R

Faţă de V 1.5 spaţiul alocat unei baze de date este structurat logic în V 1.6.R în următoarele subspaţii :

a) spaţiul fişier (FICH) : spaţiul real ataşat spaţiului virtual ce conţine datele bazei respective ;

b) spaţiul dicţionar (DICO) : conţine dicţionarele SOCRATE ataşate caracteristicilor declarate chei de acces, dicţionar pentru denumirile acestor caracteristici,

un dicționar pentru numele programelor și dicționarele asociate caracteristicilor de tip ⟨*referire*⟩. Mărimea subpaginii acestui spațiu este de 265 cuvinte ;

c) spațiul program (PROG) : rezervat stocării codului obiect direct executabil al programelor precompilate și IMT. Programele sînt paginate în pagini de 1 ko iar dimensiunea subpaginii este de 256 cuvinte ;

d) spațiul structură (STRU) : conține codul intern al structurii bazei de date, codul sursă al programelor precompilate, IMT și al macroinstrucțiunilor. Mărimea subpaginii este de 16 cuvinte ;

e) spațiul de lucru (TRAV) : manevră pentru procesoarele SOCRATE cu subpagina de 526 cuvinte. Spațiul TRAV poate fi reformatat în orice moment.

Dimensiunea fiecărui subspațiu se specifică, ca și în V 1.5 cu ajutorul comenzii % ALLOC, a cărei structură este :

%[⟨*etichetă*⟩] ALLOC FA : (⟨n_1⟩, ⟨l⟩), DA : ⟨n_2⟩, PA : ⟨n_3⟩, *
% SA : ⟨n_4⟩, WA : ⟨n_5⟩ : MA : ⟨n_6⟩

unde :

– ⟨l⟩ : dimensiunea subpaginii în spațiul FICH ;
– ⟨n_1⟩, ⟨n_2⟩, ⟨n_3⟩, ⟨n_4⟩, ⟨n_5⟩, ⟨n_6⟩ specifică numărul de cilindrii rezervați pentru :
– FA – FICH ; – DA – DICO ; – PA – PROG ;
– SA – STRU ; – WA – TRAV ; – MA – MSRT.

f) spațiul de manevră la sortare (MSRT) : dimensiunea, subpaginii 256 cuvinte.

Ca și în V 1.5. dacă n este numărul total de cilindri alocați acestor subspații atunci $(n_1 + n_2 + n_3 + n_4 + n_5 + n_6) \leqslant n$. A fost modificată structura comenzii %SYSRUN (ca parametrii acceptați) astfel :

$$\%\,[\langle \mathit{eticheta} \rangle]\ \mathrm{SYSRUN\ FN} : \left\{\begin{matrix}F\\D\\R\\V\\S\\M\end{matrix}\right\} \left[\ ,\ \mathrm{ST} : \left\{\begin{matrix}\mathrm{ALL}\\\mathrm{DICO}\\\mathrm{FICH}\\\mathrm{STRU}\\\mathrm{PROG}\\\mathrm{TRAV}\\\mathrm{MSRT}\end{matrix}\right\} \left[\ ,\ \mathrm{BS} : \left\{\begin{matrix}(\langle n\rangle, \langle m\rangle)\\(0, \mathrm{MAX})\end{matrix}\right\}\right]\right]$$

unde :

– F : formatare ;
– D : dump ;
– V : videază subpaginile bazei (sau a unui subspațiu) pe bandă magnetică ;
– R : restaurarea unei baze (sau subspațiu) pornind de la un vidaj pe bandă ;-
– S : statistici parțiale sau totale ;
– M : statistica și demarcarea paginilor marcate la utilizarea JNIB.

Spațiul (subspațiul) bazei de date care interesează este indicat prin argumentul ST :

– ALL : toate cele (6) subspații (BS interzis) ;
– unul din cele (5) subspații (BS admis).

BS : idem V.1.5 (§ 7.5.1).

Concluzii

Acest capitol a fost destinat prezentării limbajului de comandă SOCRATE. Prezentarea a fost efectuată, pe de o parte, pentru limbajul de comandă în prelucrarea conversațională, iar pe de altă parte, pentru limbajul de comandă în „batch processing".

Pentru versiunile sale de implementare sub sistemele de operare SIRIS și HELIOS limbajul de comandă pentru prelucrarea „batch processing" reprezintă o simulare a limbajului de comandă conversațional. Modul de implementare al acestei simulări este strit dependent de caracteristicile sistemului de operare gazdă. Limbajul de comandă conversațional este o caracteristică independentă de implementare, dar care, n-a fost reținută de unele implementări (Mini inclusiv). Referindu-ne la implementarea Mini aceasta nu are propriu-zis un limbaj de comandă conversațional intrinsec. Caracteristica de limbaj de comandă conversațională este imprimată de caracteristicile sistemului de operare gazdă. Mai mult majoritatea prelucrărilor au o derulare tip „batch processing", evident, în accepțiunea acestei noțiuni la sistemul de operare gazdă.

Studierea acestui capitol poate fi utilă indiferent de tipul de implementare care va fi adoptat.

Dacă ne referim la Mini, de exemplu, fișierele de comenzi prezentate la exemplificări au un echivalent în această implementare. Utilizatorul poate să-și construiască „fișiere de comenzi indirecte" specifice acestei implementări conservînd evident principiile prezentate.

7. INTEGRITATEA BAZELOR DE DATE

Generalități

Sistemul de gestiune a bazelor de date — SOCRATE — permite în mod esențial două tipuri de utilizări :

— o utilizare în mod de funcționare „batch processing" eventual prin intermediul interfeței cu limbaje evoluate ;

— o utilizare conversațională care permite utilizarea simultană a mai multor posturi de lucru (console-utilizare multiconsolă) care se adresează la aceeași bază de date sau la diferite baze de date independente (utilizare multibază). În utilizarea multibază, fiecare consolă (terminal, post) este legată, la un moment dat, numai de o singură bază de date (cea pentru care a executat procedura de conexiune LOGIN).

Deoarece sistemul admite accesul simultan al mai multor console (posturi de lucru) la aceeași bază de date iar limbajul de interogare, pus la dispoziție de sistemul SOCRATE, permite atît extragerea cît și modificarea datelor conținute de o bază de

date, există posibilitatea, indiferent de gradul de automatizare și sofistificare al procedurilor puse în lucru, alterării sau distrugerii accidentale a informațiilor. Mai mult, chiar dacă nu există nici un pericol real de distrugere sau alterare a informațiilor prin funcțiunile specifice ale sistemului sau cele programate de utilizator, există, totuși, factorii externi sistemului (căderi ale sistemului datorate defecțiunilor hard, întrerupere sursa de alimentare electrică etc.) care pot distruge baza de date, sau datele din baza de date, fizic sau logic (din punct de vedere al semnificației logice a datelor conținute în bază).

Soluția conservării informației, ca măsură de prevedere contra distrugerii, se alege în funcție de criterii de eficiență și de felul distrugerii :

1. prevederea distrugerii fizice a suportului bazei de date se face prin dublarea suportului pe un suport identic sau diferit. Această dublare este obligatorie ;

2. prevederea distrugerii logice a informațiilor se poate realiza prin :

— salvarea conținutului bazei de date, după un punct de referință ales de utilizator conform unor criterii care se referă la timp de prelucrare, număr de actualizări efectuate etc. Conținutul bazei de date este considerat coerent la acest punct de referință (și trebuie să fie coerent). Soluția este eficientă în cazul bazelor de date de volum mic ;

— salvarea conținutului bazei de date la un moment dat, considerat moment de referință, iar din acest moment se construiește un jurnal (fișier în care se conservă informațiile care suferă modificări în timp) care să permită aducerea la zi a bazei de date salvate anterior.

Sistemul SOCRATE constituie, la cerere, două fișiere jurnal care ne permit să conservăm informațiile unei (unor) baze de date începînd cu un punct de control, ales de utilizator în stadiul în care baza sa de date se prezintă într-o stare coerentă, astfel :

— diferențialul sau jurnalul „after" (JNLA) care înregistrează modificările aduse conținutului bazei de date (puneri la zi, suprimări, adăugări etc.) ;

— diferențialul invers sau jurnalul „before" (JNLB) care înregistrează datele conținute într-un cuantum de informație (pagina) înainte de a o modifica pentru prima dată după un punct de referință ales de utilizator.

Opțiunea de utilizare, sau nu, a procedurilor de securitate a fost lăsată la latitudinea utilizatorului deoarece nu toate sesiunile de lucru necesită utilizarea fișierelor jurnal (ex. extrageri de date), utilizare care presupune un surplus de spațiu de memorie și timp de prelucrare.

Punct de referință

Este punctul de plecare ales de utilizator pentru a înregistra în JNLB paginile din baza (bazele) de date, activă(e) în acel moment înainte de prima modificare și de a le marca în baza (bazele) de date de proveniență. Marcarea se realizează prin poziționarea unui indicator de recopiere, reprezentat de obicei la nivel de bit (atribuirea valorii 1 bitului asociat).

Pentru construirea unui punct de referință se fac precizările :

— baza (bazele) trebuie să fie într-o stare coerentă din punct de vedere fizic (înlănțuire corectă) și din punct de vedere logic (informații corecte din punct de vedere al utilizatorului) ;

— construirea efectivă se face la inițiativa utilizatorului la un moment care poate fi :

• sfîrșit corect de sesiune de lucru ;
• după o restaurare a bazei de date.

Stabilirea stării de coerență în care se află o bază de date se face prin :

— rularea utilizatorului de statistici (verificare coerență fizică) ;

— rularea programelor de test pentru determinarea coerenței logice.

După construirea punctului de referință sînt obligatorii următoarele operații :

— conservarea și numerotarea cronologică a benzilor de securitate ;

— conservarea, pentru fiecare tip de restaurare posibilă, a numărului CKPT „sfîrșit de sesiune".

Punct de control (CKPT-ChecKPoint Triggering)

Este un punct în care baza de date trebuie să fie coerentă (fizic și/sau logic) a cărui înregistrare pe suportul extern se face la momentul îndeplinirii condițiilor specificate de utilizator. Revenirea la un punct de control definit anterior, se face prin readucerea bazei în starea prezentată în punctul de referință (prin utilizarea JNLB) apoi se adaugă modificările pînă la punctul de control (utilizarea JNLA). Declanșarea înregistrării unui punct de control presupune o oprire a programelor aflate în execuție, oprire care cere acordul tuturor utilizatorilor simultani al bazei (bazelor) de date (acordul se dă într-o stare semantică coerentă a bazei de date).

Numerotarea punctelor de control începe cu 0, iar declanșarea CKPT se face automat la lansarea sesiunii de lucru.

Declanșarea următoarelor puncte de control, din cadrul sesiunii de lucru, depinde de tipul de sesiune, astfel :

– „batch processing" : validarea declanșării punctelor de control se realizează prin prezența comenzii %SAVE. La acest tip de sesiune nu are loc declanșarea unui punct de control la LOGOUT iar închiderea benzii de securitate are loc prin comandaCALL SSAVE ;

– interfața COBOL : declanșarea are loc la comanda CALL SSAVE. În cadrul acestui tip de sesiune, se declanșează automat CKPT la fiecare LOGOUT executat pe oricare din bazele active. Închiderea în sens SOCRATE a benzii la sfîrșit de sesiune se realizează o singură dată pentru toate bazele închise sau care au fost obiectul unei comenzi CALL STERM ;

– conversațional : declanșarea are loc la :

• comanda operatorului central ;
• trecerea la 0 a unui contor de citire și/sau scriere pe baza de date (valoarea contorului este stabilită de utilizator și poate fi modificată dinamic) ;
• comanda LOGOUT emisă de un utilizator.

Înregistrarea punctelor de control

Declanșarea operației de constituire a punctului de control echivalează cu o întrerupere a sistemului, întrerupere care trebuie să fie tratată atunci cînd baza (bazele de date) activă a atins o stare coerentă în sens :

– fizic (înlănțuire corectă a subpaginilor) ;

— semantic (punerea la zi a unei caracteristici elementare terminată) ;

— logic (atingerea (terminarea) unui ansamblu de puneri la zi cuprins între BLOQUER și LIBERER).

Evenimentele care pot să declanșeze înregistrarea punctelor de control sînt :

— comanda operatorului central ;

— trecerea la 0 a unui contor de punere la zi specificat de utilizator (sau operator central) la momentul lansării sesiunii de lucru (sau a lucrului său) sau pe parcursul desfășurării acesteia.

La atingerea unui eveniment de declanșare a unui punct de control se transmit următoarele mesaje :

— CKPT DUE... — la consola operatorului central ;

— PAUSE EN FIN DE REQUET — la toate posturile active ;

— CKPT EN COURS — ca răspuns la orice tentativă de LOGIN.

În acest moment, operatorii de la terminale, tastează la inițiativa lor comanda PAUSE care provoacă blocarea tastaturii terminalului pînă la declanșarea efectivă a CKPT. Fiecare comandă PAUSE este indicată la consola operatorului central prin mesajul :

LN : TLxx TE : YY PAUSE este :

— xx identifică linia ;

— yy identifică terminalul și rangul său în lista de selecție.

După efectuarea comenzii PAUSE, pe toate consolele active, are loc declanșarea efectivă a CKPT cerut. Acest CKPT este anunțat (numărul său) atît la consola operatorului central cît și la posturile active.

În cursul efectuării operației de constituire a CKPT SOCRATE asigură, în mod automat, coerența fizică a bazei de date ; coerența logică a bazei de date va fi asigurată de către utilizator (prin alegerea momentului de emitere a comenzii PAUSE).

Deoarece comanda PAUSE provoacă o blocare a oricărei acțiuni asupra bazei de date, la care era conectată, operatorul de la terminal poate verifica starea liniei de transmisie prin trimiterea unor „mesaje“ la care sistemul răspunde cu ? (de exemplu, comenzi eronate).

Deoarece sistemul funcționează la modul „multi-bază“-„multi-consolă“, este posibilă producerea unui incident după cererea de deconexiune a unui utilizator, incident care poate cere o reluare de la un punct de control anterior momentului terminării lucrului său. Pentru a conserva lucrul efectuat de operatorii de la terminale, poate fi prevăzută următoarea disciplină de lucru :

— înainte de efectuarea cererii de deconexiune (LOGOUT) utilizatorul semnalează acest lucru operatorului central prin comanda MESSOP ;

— operatorul central cere, în acest moment, construirea unui punct de control (prin CKPT) ;

— după constituirea CKPT cerut, utilizatorul poate efectua cererea sa de deconexiune.

Jurnalul A

Jurnalul A reprezintă un „diferențial“ între starea bazei de date la începutul sesiunii de lucru (care poate fi, dar nu neapărat, punct de referință) și starea sa la sfîrșitul sesiunii. Construirea bazelor identice, în momentul în care volumul diferențialului este destul de mare, este lăsată la latitudinea utilizatorului.

Pentru a explica modul de constituire a fișierului jurnal A considerăm o bază de date, B_0 și copia sa B_0'. Modul de constituire a fișierului jurnal și de aducere a unei baze de date, pe baza lui, la zi este următorul:

- B_0 este utilizată în timpul sesiunii de lucru 1 și suferă modificări care o aduc în starea B_1.SGBD-SOCRATE a construit, în acest caz, fișierul jurnal A (ΔB_1) astfel încît realizînd „intersecția" între B_0' și ΔB_1 obținem o bază B_1' care este identică cu B_1;
- la o nouă utilizare a lui B_1, pentru a deveni B_2 obținem fișierul jurnal cu ΔB_2 astfel încît:

$$(B_1' \wedge \Delta B_2) \rightarrow B_2;$$

— generalizînd, se poate constata că dacă utilizăm baza B_0 în sesiunile 1, 2, 3...n pentru fiecare sesiune obținem diferențialele ΔB_1, ΔB_2, $\Delta B_3 \ldots, \Delta B_n$, astfel încît realizînd intersecțiile, între aceste diferențiale și baza B_0', în ordinea definită de ordinea temporală a producerii lor, putem obține o imagine a bazei de date în oricare din stările anterioare sau o bază identică cu cea din starea curentă, la care am ajuns.

În condițiile expuse intersecția poate fi generalizată astfel:

$$\Delta B_n \wedge (\ldots \wedge (\Delta B_3 \wedge (\Delta B_2 \wedge (\Delta B_1 \wedge B_0'))) \ldots) \rightarrow B_n$$

Operația de intersecție ($\wedge$) între o bază de date B și un diferențial asociat ΔB reprezintă înlocuirea informațiilor din baza de date B cu informațiile corespondente (care au aceeași adresă) din diferențialul ΔB.

Acest mod de lucru permite, deci, ca plecînd de la o bază de date, considerată într-o stare inițială luată ca nivel de referință (punct), să obținem o bază de date identică cu cea cu care lucrăm, sau să obținem orice stare anterioară a bazei de date, stare anterioară definită între nivelul de referință considerat și nivelul actual al bazei de date.

Jurnalul B

Vom trata acum modul de conservare a informației în cazul în care pe parcursul desfășurării unei sesiuni de lucru avem un incident care aduce datele din baza de date într-o stare incoerentă. În acest caz, soluția aplicării operației de constituire a stării anterioare a bazei de date (stare considerată coerentă) prin intersecția diferențialelor cu baza de date de referință poate fi costisitoare. Acest lucru a dus la definirea unei proceduri de lucru care să permită aducerea unei baze în starea anterioară producerii incidentului pornind de la stare curentă. Pentru realizarea acestei operații, SGBD-SOCRATE construiește la cerere „diferențialul invers" care permite, prin realizarea unei operații de intersecție, similară cu cea descrisă la JNLA, să obținem, pe același suport, o imagine anterioară (deci o stare anterioară) a bazei de date.

Considerăm o bază B_0 și copia sa B_0'; baza B_0 este utilizată în cursul sesiunii de lucru și devine B_1. În acest caz SGBD-SOCRATE constituie:

— JNLA care conține ΔB_1 astfel încît $\Delta B_1 \wedge B_0' \rightarrow B_1$

— JNLB care conține ΔB_{1I} astfel încît $B_1 \wedge \Delta B_{1I} \rightarrow B_0 = B_0'$ unde prin ΔB_{1J} am notat diferențialul invers din momentul j.

Din modul de definire a intersecțiilor rezultă că aplicînd succesiv intersecțiile lui B_1 cu cele două fișiere jurnal obținem tot B_1 adică : $(B \wedge \Delta B_{i1}) \wedge \Delta B_1 \rightarrow B_1$.

Aparent, utilizarea celor două tipuri de jurnal pare redondantă.

Jurnalul A presupune efectuarea unor prelucrări de volum mare în timp, prelucrări care determină o creștere considerabilă a volumului acestui fișier. Aducerea la zi a copiei bazei de date se face la momente de timp distanțate (ca timp calendaristic efectiv sau ca volum de prelucrări) și numai dacă este necesar. Intersecția acestui jurnal A cu baza inițială pentru a o aduce în starea anterioară producerii incoerenței este ineficientă (datorită timpilor de prelucrare). Acest sistem ar fi foarte greoi mai ales în cazul în care volumul prelucrărilor este foarte mare și necesită puncte de reluare. Din acest motiv sistemul pune la dispoziție procedura definită pentru jurnalul B. În cazul în care volumul prelucrărilor unei sesiuni de lucru este important, devine indispensabil modul de lucru cu puncte de reluare. Definirea punctelor de reluare se face împărțind sesiunea în „mini-sesiuni", o „mini-sesiune" reprezentînd în acest caz un punct de reluare. Dacă dorim, la acest mod de lucru, să utilizăm, pentru reluare în caz de incident, fișierul jurnal A, atunci acest lucru presupune să avem două baze de date identice la începutul sesiunii și organizate pe același tip de suport, ceea ce poate fi o penalizare foarte mare a resurselor externe ale calculatorului iar riscul de a distruge baza de date (și copia) crește. Utilizarea jurnalului B ne dă posibilitatea în acest caz, să reconstruim baza de date, aflată pe suportul de lucru activ, în orice stare din cadrul sesiunii (stare corespunzătoare unui punct de reluare definit la nivel de minisesiune). Timpul de reluare devine incompatibil mai mic decît cel necesar la reluare cu utilizarea jurnalului A.

Vom detalia modul de lucru în cazul în care sesiunea este divizată în „mini-sesiuni". Presupunem că avem o bază de date aflată în starea B_0 care în decursul sesiunii ia diverse stări de $b_1, b_2, \ldots, b_n = B_1$, fiecare stare corespunzînd unei mini-sesiuni.

Diferențialele realizate în acest caz sînt $\Delta b_1, \Delta b_2, \ldots, \Delta b_n$ unde Δb_i reprezintă modificările care permit trecerea de la o stare b_{i-1} la starea b_i. În acest caz, diferențialele pot conține mai multe înregistrări care reprezintă modificarea succesivă a aceleeași informații deoarece nu se știe la momentul t (corespunzător mini-sesiunii de la momentul t) că informația va mai fi modificată la momentul $t + p$ (mini-sesiunea de la momentul $t + p$).

Diferențialele inverse realizate sînt $\Delta b_{11}, \Delta b_{12}, \ldots, \Delta b_{1n}$, iar intersecția lor cu baza curentă pot da baza de plecare B_0. Realizarea acestui diferențial constă în protejarea informației înainte de prima modificare în cazul sesiunii. Orice modificare ulterioară a informației în cazul aceleiași sesiuni, nu este înregistrată în jurnal. Rezultă deci că la sfîrșitul fiecărei sesiuni putem aduce baza, prin intersecție cu JNLB, în starea în care se află la începutul sesiunii.

Schematic, modul de construire a fișierelor jurnal în condițiile expuse anterior, se prezintă ca în figura 7.1.

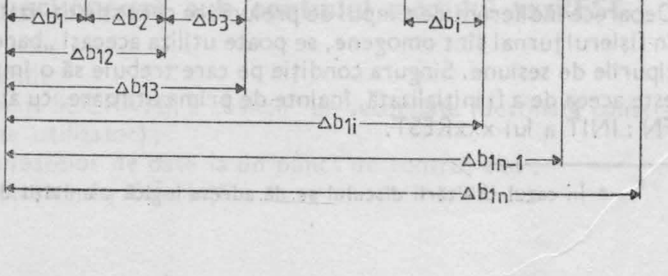


Fig. 7.1. Construirea fișierelor jurnal în cazul utilizării mini-sesiunilor.

Dacă avem, de exemplu, un incident între starea b_2 și starea b_3 se va realiza intersecția $(\Delta b_{13} \wedge b_3) \rightarrow b_2$ și se reia de la starea b_2.

În implementarea Mini și V 1.6.R. există posibilitatea întoarcerii la ultimul „punct de control" definit prin intermediul unui jurnal rapid (JNLQ) construit pe disc. Modul de funcționare al acestui jurnal este de tipul jurnalului B, putînd fi definit ca o restricție a acestuia.

De altfel, acest tip de jurnal „quick" este singurul implementat pe Mini, cu avantajele și dezavantajele sale.

O implementare eficientă a jurnalelor este cea realizată pentru V 1.5 la C.S.P. în sensul că suportul jurnalului poate fi discul magnetic ceea ce face ca utilizarea de tip jurnal „quick" să fie combinată cu avantajele oferite de utilizarea lui JNLA și JNLB.

Cererea de constituire a jurnalelor A și B

Pentru a putea utiliza o bandă de securitate (inițializată apriori cu SOCREST), este necesar ca la cererea de apel a unui modul de prelucrare (batch, caracter, mesaj sau interfață) să se definească în sens S.G.F. fișierul jurnal (cu ajutorul comenzilor .LABEL și .ASSIGN) pentru care se impune utilizarea indexului de exploatare Z. În afara acestei definiri, este necesar să se rezerve, în comanda .OPTION, o zonă statică de cel puțin 2 400 octeți și să se specifice tipul/tipurile de jurnal care se va/vor constitui (parametrul SF, vezi comanda .OPTION).

Structura minimală a comenzilor de lansare a unei sesiuni utilizînd fișierul de securitate este :

```
    . XPROC PROGEN,MOD
    . MOD &MODUL: <modul>&
    . * <modul> ::=SOCBTCH/SOCMULC/SOCMULM
    . ENDMOD
1 > . ASSIGN Z, DV:MT**
2 > . LABEL Z,FN: '<nume-jurnal>'
3 > . OPTION CS,CF:10, DS:2400, SF: ([JNLA] [,JNLB])'
    . RUN TIME:999,NL:50000,AD:0,0
  [ % SOC BN:BDDPERS,AN:1,PN:AVRAM.PW:SOC  ]
  [ <apel procesor/procesoare prelucrare>  ]
  [ % LOGOUT                               ]
```

MT : adresa logică a unității de „bandă magnetică"* pe care este montat fișierul jurnal ;**
⟨*nume-jurnal*⟩ : numele dat fișierului jurnal la inițializarea „benzii" jurnal ;
SF : cererea de constituire a fișierului jurnal.

La utilizarea interfeței este suficientă plasarea comenzilor 1⟩, 2⟩ și 3⟩ înaintea comenzii .RUN.
Deoarece indiferent de timpul de prelucrare cerut de utilizator, înregistrările înscrise în fișierul jurnal sînt omogene, se poate utiliza aceeași „bandă" de securitate în toate tipurile de sesiune. Singura condiție pe care trebuie să o împlinească această „bandă" este aceea de a fi inițializată, înainte de prima utilizare, cu ajutorul funcției %MAINT FN : INIT a lui xxxREST.

---

* În cazul utilizării discului se dă adresa logică a unității de disc

Reluări

Modurile de reluare diferă în funcție de starea în care este adusă baza de date și se pot realiza prin :

1) retur de sfîrșit de sesiune ;
2) retur la un punct de referință de plecare ;
3) retur la un punct de control explicit ;
4) retur la început de sesiune.

1) **Retur la sfîrșit de sesiune** : presupune utilizarea unei copii a bazei de date care va fi adusă în starea curentă. Acest tip de reluare este indicat în cazul în care sesiunea (sesiunile) derulată(e) anterior s-a(u) terminat normal după care, s-a produs o alterare fizică a suportului. Reluarea se face exploatînd JNLA, constituit după punctul de referință definit de utilizator, exploatare care realizează includerea modificărilor, ulterioare acestui punct în copia bazei de date. Prin această operație de „aducere la zi" a unei copii anterioare nu se modifică nici indicatorul de recopiere a paginilor marcate și nici punctul de referință.

2) **Retur la un punct de referință de plecare** : se realizează prin exploatarea JNLB, obținut după acest punct de referință, exploatare care realizează ștergerea indicatorilor de recopiere aferenți paginilor restaurate (paginile prezentate în JNLB, aferente bazei de date prelucrate). Dacă se dorește aducerea la zi a acestei baze (din starea în care se afla la momentul definirii punctului de referință) atunci este necesar să se refacă toate punerile la zi derulate între momentul definirii punctului de referință și momentul în care ne aflăm.

3) **Retur la un punct de control explicit** : se realizează prin exploatarea JNLA și JNLB obținute după punctul de referință definit pînă la punctul de control după care se va exploata numai JNLB obținut după punctul de control ales. Operațiile efectuate de utilitarul de restaurare, la acest tip de restaurare, sînt :

a) între începutul „benzii" (benzilor) de securitate și punctul de control :
— restaurarea paginilor JNLB (indicatorul de recopiere este lăsat la valoarea 1) ;
— copierea modificărilor JNLA ;

b) după punctul de control : recopierea paginilor JNLB cu punerea la 0 a indicatorului de recopiere ; această recopiere antrenează o anulare a punerilor la zi efectuate după acest punct.

Executarea acestui tip de reluare presupune utilizarea bazei de date aflate în lucru și „banda" (benzile) de securitate (în ordinea constituirii lor) constituită(e), anterior sesiunii în care s-a definit punctul de control.

4) **Retur la început de sesiune** : se utilizează atunci cînd în cursul unei sesiuni, avem un incident fatal înainte de crearea primului punct de referință.

Se utilizează baza de date aflată în starea curentă și fișierul jurnal constituit în decursul sesiunii cu incidentul.

Utilitare care funcționează sub controlul modului xxxREST

Aceste utilitare permit :

— inițializarea, în sens SOCRATE, a „benzii" de securitate (destinată constituirii jurnalelor cerute de utilizator) ;
— restaurarea bazei/bazelor de date la un punct de control ales ;
— redefinirea punctelor de referință.

Inițializarea unui fișier de securitate

Structura generală a liniilor de comandă necesare inițializării (SIRIS/HELIOS) este următoarea:

```
.    XPROC PROGEN,MOD
.    MOD &PROC:SOCREST &
.    ENDMOD
.    OPTION CS'CF:10,DS:2500'
.    RUN TIME:999,NL:500000,AD:0,0
$
%    SOC BN:SOCRATE
%    ATTACH JRNL
%    LABEL FN: '⟨nume-jurnal⟩'
%    ASSIGN DV:MT**
%    FILE PRM:OUT,RCS:1100,BFS:1100,RCF:VAR
%    MAINT [FN:INIT]
%    LOGOUT
```

⟨*nume-jurnal*⟩ : = numele acordat fișierului jurnal ;
MT**:: = unitatea logică de „bandă magnetică" pe care se află „banda" cu fișierul jurnal.

Parametrul FN : INIT cere inițializarea „benzii" de securitate ; în cazul în care argumentul lipsește va fi cerut la consola sistemului de calcul sub forma :
pp SOCRATE**MAINT
La această cerere operatorul răspunde :
FN : INIT ⟨cr⟩, validînd inițializarea.

Restaurarea la un punct de reluare

Secvența liniilor de comandă necesare restaurării (SIRIS/HELIOS) este următoarea :

```
.    XPROC PROCGEN,MOD
.    MOD &PROC:SOCREST &
.    ENDMOD
.    OPTION CS'CF:10,DS:2500'
.    RUN TIME:999,NL:500000,AD:0,0
$
%    SOC BN:SOCRATE
%    ATTACH JRNL
%    LABEL FN: '⟨nume-jurnal⟩'
%    ASSIGN DV:MT**
%    FILE PRM:INP, RCS:1100,BFS:1100,RCF:VAR
%    MAINT [FN: {RES|NCL}, LC:(n,m) [,ALL]]
[linii cu numele bazelor pe care le tratăm (1 linie/1 nume bază)]
%    LOGOUT
```

Semnificația parametrilor comenzii MAINT este :

— RES : restaurare cu demarcare pînă la punctul definit de parametrul LC. Procesorul realizează o marcare a paginilor înregistrate în JNLB pînă la punctul de reluare definit care devine în acest caz punct de referință ;
— NCL : restaurare fără demarcare ;
— LC : permite definirea punctului de reluare ales prin :
n : numărul CKPT (checkpoint triggering) ales ;
m : numărul blocului ales după CKPT ;

— ALL : specifică faptul că se efectuează restaurarea tuturor bazelor care au utilizat fișierul jurnal considerat. Dacă acest argument este absent atunci comanda %MAINT trebuie să fie urmată de linii cu numele tuturor bazelor tratate, astfel :

% MAINT BAZA 1 BAZA 2 . . . BAZA 20 EOF	Lista bazelor de date trebuie să se termine printr-o comandă EOF, iar numărul maxim de baze admise este de 20.

Dacă utilizatorul nu specifică parametrii comenzii % MAINT, atunci sistemul cere lista argumentelor la consola sistemului prin :

pp SOCRATE** MAINT iar operatorul poate răspunde :

FN : —,LC: (—,—) [, ALL] (NL)

Dacă argumentul ALL nu este prezent sistemul cere lista bazelor de date :

* ⟨*nume bază*⟩ ⟨cr⟩

.

. (max. 20 baze)

.

* EOF ⟨cr⟩

unde :

* este promptul prin care sistemul anunță că așteaptă introducerea unei linii cu numele unei baze de date sau terminatorului. EOF care anunță că lista s-a încheiat.

Modul de funcționare a utilitarului de restaurare la o furnizare corectă a argumentelor, este următorul :

— *„banda"*, care conține fișierul jurnal, este citită pînă la sfîrșitul fișierului jurnal, indiferent de numărul de CKPT cerut, pentru a recopia, eventual, paginile din JNLB a bazei (bazelor) în curs de restaurare ;

— dacă fișierul jurnal nu a fost închis, în sens SOCRATE, utilitarul editează un mesaj de atenționare la consola sistemului ;

— dacă este găsit CKPT cerut, sistemul înscrie după el două înregistrări : *„sfîrșit de sesiune"* și *„sfîrșit fișier SOCRATE"* iar fișierul este închis în sens SGF. Înregistrările posterioare CKPT cerute vor fi pierdute. Scrierea acestor înregistrări speciale asigură o închidere a „benzii", închidere care ne va permite să o reutilizăm într-o sesiune următoare ;

— dacă nu este cerută restaurarea tuturor bazelor active la momentul CKPT cerut sau dacă există înregistrări care se referă la baze care n-au fost cerute, posterioare CKPT cerut, utilitarul de restaurare va efectua, în mod automat, restaurarea bazelor „unitate" și le va semnala la consola sistemului.

Dacă, la începutul primei sesiuni, toate bazele sînt într-o stare de referință (nici-o pagină marcată) atunci primul CKPT al primei sesiuni va fi considerat punct de referință. În cazul în care dorim să ne întoarcem la primul punct de referință, parametrul LC al comenzii MAINT are valoarea LC: (0,2) ia RES și NCL sînt echivalente.

Pentru punerea la zi a unei copii a bazei de date (utilizare JNLA), structura comenzi MAINT este :

$$\% \quad \text{MAINT FN:} \begin{Bmatrix} \text{RES} \\ \text{NCL} \end{Bmatrix}$$

Redefinirea punctelor de referință

Marcarea paginilor bazei (bazelor) de date, înregistrate în JNLB, se face în scopul evitării recopierilor multiple ale acestor pagini. Sistemul SOCRATE permite construirea dinamică a punctelor de referință ale bazei de date prin demarcarea paginilor acestora, memorate în JNLB, completînd comanda MAINT astfel :

% MAINT FN:CLR [,ALL]
[⟨*nume bază*⟩]
EOF

Decizia de demarcare a paginilor bazei (bazelor) de date se ia în momentul în care baza (bazele) este considerată într-o stare fizică coerentă. După realizarea demarcării, „banda" care conține fișierul jurnal, construit pînă în momentul demarcării, poate fi inițializată (sau se inițializează o altă „bandă" de securitate) pentru a construi un nou fișier jurnal.

Mesaje de eroare

La furnizarea incorectă a unor parametri ceruți de utilitar sau la detectarea unor erori de coerență a datelor tratate se emite un mesaj de eroare. Mesajele care pot apărea sînt :

DISQUE ABSENT : răspuns ⟨cr⟩ pe cererea de afectare a unei unități de disc ;
BASE INCONUE : eroare în curs de LOGIN ;
* * * CKPT POSTERIEUR A CELUI DEFINI PAR LC : întîlnirea unui 'CKPT' posterior celui cerut ;
CKPT DEFINI PAR LC NON TRUVE : CKPT specificat eronat ;
ERREUR LECTURE T.M. : fișierul jurnal a fost închis în sens SGF dar nu și în sens SOCRATE (se tratează banda cu modulul FERME, după care se reia prelucrarea) ;
ERREUR COMPARAISON INFOS DE CTRL : a fost atins sfîrșitul fizic al benzii jurnal și nu au fost detectate informațiile de închidere logică (banda este neînchisă în sens SOCRATE și SGF) ;
TYPE D'ENREG ERRONE : înregistrare eronată
ERREUR E / S n_0-BLOC-n_0 : idem precedent ;
ERREUR SUR SEQUENCE DE CKPT : banda jurnal incoerentă ;
TROP DE BASES : au fost definite mai mult de 20 de baze ;
BASES A RESTAURER :
⟨*nume bază*⟩ *
mai sînt baze de restaurat deoarece n-au fost toate cerute la activarea CKPT ;
FIN DE RESTAURATION : sfîrșit normal de restaurare.

Indiferent de modul de terminare a sesiunilor precedente restaurării, utilitarul realizează închiderea benzii jurnal în sens SOCRATE și SGF făcînd-o aptă pentru utilizarea într-o sesiune viitoare.

În V 1.6.R. tentativa de lansare în execuție a unei prelucrări pe o bază de date care n-a fost refăcută, după producerea unui incident în cazul unei sesiuni care utiliza securitatea, provoacă mesajul:

$$\text{ESPACE}\left\{\begin{array}{l}\text{FICH}\\ \text{DICO}\\ \text{PROG}\\ \text{STRU}\end{array}\right\}\text{INCOERENT} - \text{ABANDON DE LA SESION}$$

Dacă spațiul TRAV sau MSRT este marcat ca eronat atunci se antrenează, în mod automat, procesul de formatare al acestuia.

Concluzii

Acest capitol este destinat tratării modului de refacere a consistenței bazei (bazelor) de date aflată(e) în exploatare la momentul producerii unui incident grav pe calculatorul care le gestionează. Metoda de refacere este utilă în egală măsură și în momentul în care un utilizator oarecare produce o alterare (in)voluntară, prin algoritmii incluși în programele sale, a bazei de date. Metoda propusă se bazează pe utilizarea unor rutine de jurnalizare incorporate în software-ul suport și apelabile parametric.

Aceste paragrafe au fost incluse în cadrul prezentării generale deoarece această implementare este singura de la noi care se încadrează în teoria generală. La implementarea Mini reluarea prelucrărilor este bazată numai pe jurnal de tip rapid („quick"), evident cu dezavantajele și avantajele acestei utilizări.

Tehnicile generale de jurnalizare și refacere pot fi adaptate și la alte sisteme informatice, care nu utilizează SGBD-SOCRATE, dar în acest caz implementarea lor revine utilizatorului. În acest context remarcăm faptul că deși problema alterării informațiilor nu este specifică bazelor de date ci aparține prelucrării automate cu „fișiere", SGBD-ul prezentat de noi încorporează tehnici de refacere și jurnalizare foarte eficiente și foarte ușor de folosit de către utilizatori.

Optimizări şi dezvoltări de SGBD-uri

8. OPTIMIZAREA STRUCTURILOR BAZELOR DE DATE ŞI A PROGRAMELOR DE EXPLOATARE

Problema optimizării structurilor bazelor de date este foarte complexă implicînd aplicarea unor tehnici speciale în toate etapele de construire a unei baze de date.

Etapele construirii unei baze de date pot fi privite ca în figura 8.1.

Etapele automatizate presupun existenţa unui SGBD a cărui alegere se efectuează, de obicei, analizînd următoarele criterii:

a) modul de descriere a informaţiilor şi relaţiilor dintre ele;

b) mijloacele de acces la informaţii (căutarea în structură);

c) structura datelor pe suportul fizic.

Dacă la analiza SGBD-ului acordăm o importanţă mai mare unuia sau altuia din aceste criterii putem obţine, pentru aceeaşi aplicaţie, baze de date care pot avea structuri diferite.

Majoritatea modelelor de SGBD-uri (ierarhice şi derivatele acestora-reţea) îl fac pe utilizator să insiste mai mult asupra ultimelor două criterii neglijîndu-l pe primul. Modelul relaţional îl degrevează pe utilizator de aceste două aspecte făcîndu-l să insiste mai mult asupra primului (a).

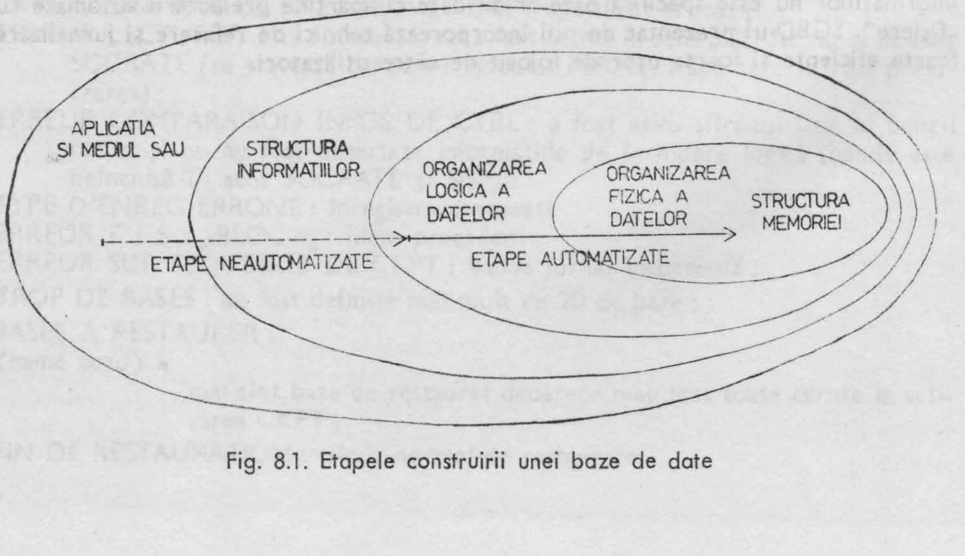


Fig. 8.1. Etapele construirii unei baze de date

Problema de optimizare pe care o propunem este de a analiza aceste criterii în ansamblu realizînd pentru fiecare optimizările posibile.

Obiectivele acestei optimizări sînt următoarele :

— asigurarea unei independențe sporite a structurii logice a datelor față de structura fizică ;

— eliminarea anomaliilor de actualizare, modificare, inserare, ștergere a datelor ;

— reducerea timpilor de răspuns la întrebările adresate bazei de date ;

— reducerea spațiului fizic ocupat pe suportul extern.

Pentru realizarea acestor obiective propunem parcurgerea următorilor pași :

1) analiza domeniului real pentru investigarea entităților, relațiilor și atributelor ; obținerea schemei externe și conceptuale. Se obține structura bazei de date într-o formă inițială ;

2) utilizarea teoriei normalizării pentru a transpune schema obținută la pasul 1 într-o schemă relațională ;

3) transpunerea schemei relaționale obținută la pasul 2 într-o schemă conceptuală de un alt tip de SGBD ;

4) optimizarea structurii astfel obținute din punct de vedere al timpului de acces și al spațiului fizic ocupat.

1) În acest pas analistul investighează domeniul real și determină un model al datelor, model care trebuie să asigure necesarul de informații pentru toate aplicațiile funcționale pe care trebuie să le satisfacă sistemul informatic.

Parcurgerea acestui pas presupune utilizarea unor tehnici manuale, semiautomate sau automate prin care se determină informațiile care vor constitui structura bazei de date.

În această structură relațiile stabilite între cîmpurile elementare („Numim cîmp elementar X o caracteristică a unui fapt sau obiect despre care dorim să înregistrăm informații. Un cîmp elementar este un ansamblu de valori asociate cîmpului numite date [D1]) pot fi de tipul semantic sau logic.

Dacă A și B sînt două cîmpuri elementare [S3] atunci :

— spunînd că A și B verifică **relația semantică** $\overline{R}$ indicăm numai faptul că asocierea acestor cîmpuri are sens din punct de vedere al aplicației considerate. O astfel de relație este simetrică : $\forall A,B \rightarrow A\overline{R}B \Rightarrow B\overline{R}A$;

— spunînd A și B (în această ordine) verifică **relația logică** $\overline{R}$ înseamnă că B este un cîmp descendent al lui A. O astfel de relație nu este simetrică : $\forall A,B \rightarrow$ $\rightarrow [A\overline{R}B] \Rightarrow$ non $[B\overline{R}A]$.

În această structură alegerea unui model de parcurgere (căi de acces) ne arată că prin definirea relațiilor la nivel logic sîntem implicați într-o structură iar prin definirea relațiilor semantice obținem o reprezentare ideală cu toate gradele de libertate posibile structură în care va trebui să alegem întotdeauna între $[A \rightarrow B]$ și $[B \rightarrow A]$. Entitățile identificate, împreună cu atributele lor se vor reprezenta sub forma de **relații** specifice bazelor de date relaționale.

2) În acest pas se va aplica teoria normalizării, specifică bazelor de date relaționale, pentru a elimina anomaliile de actualizare, inserare, modificare sau ștergere a datelor.

Aplicarea normalizării duce la obținerea unei scheme relaționale a bazei de date, schemă care poate fi descrisă cu ajutorul LDD al altui tip de SGBD, optimă din punct de vedere al rezultatului corect al operațiilor care se vor efectua asupra bazei de date.

Modul de realizare a normalizării [CODD] este :

FN1 :: transpunerea schemei relaţionale în FN1 :

0. se descompune schema în scheme arborescente ;
1. se ia prima relaţie de la rădăcina arborelui iar cheia sa primară se inserează înaintea cheii primare a relaţiilor subordonate ;
2. se suprimă în relaţia rădăcină cîmpurile agregate care acceptă descompunerea pînă la nivel elementar ; relaţia este proiectată în atîtea relaţii suplimentare cîte cîmpuri repetitive există ;
3. se elimină această relaţie din arbore rezultînd sub-arbori ;
4. se repetă paşii 1 – 3 pentru fiecare sub-arbore al structurii.

Definiţie :

„O relaţie R este în FN1 dacă şi numai dacă toate atributele conţin numai valori atomice".

FN2 :: transpunerea schemei relaţionale în FN2 :

1. se ia o relaţie şi se determină dependenţele funcţionale dintre atribute (dependenţe funcţionale complete şi parţiale) ;
2. **dacă** relaţia posedă dependenţe funcţionale parţiale
   atunci se descompune (prin proiecţie) relaţia în relaţii cu dependenţe funcţionale complete ;
   altfel relaţia este în FN2 ;
3. se repetă paşii 1 – 2 pentru fiecare relaţie a structurii ;

Definiţie :

„O relaţie R este în FN2 dacă este în FN1 şi dacă fiecare atribut din relaţie este dependent funcţional complet de fiecare cheie posibilă a relaţiei".

FN3 :: transpunerea schemei relaţionale în FN3 :

1. se ia o relaţie şi **se determină dependenţele tranzitive** ale atributelor faţă de cheia relaţiei ;
2. **dacă** relaţia posedă dependenţe tranzitive
   atunci se descompune (prin proiecţie) relaţia în relaţii care nu conţin dependenţe tranzitive ;
   altfel relaţia este în FN3 ;
3. se repetă 1 – 2 pentru fiecare relaţie a structurii ;

Definiţie :

„O relaţie este în FN3 dacă ea este în FN2 şi dacă fiecare atribut neprim al relaţiei depinde în mod netranzitiv de fiecare cheie posibilă a relaţiei".

FN4 :: transpunerea schemei în FN4 :

1. se ia o relaţie şi **se determină dependenţele multivaloare** ;
2. **dacă** relaţia posedă dependenţe multivaloare **atunci** relaţia se descompune (prin proiecţie) în relaţii care nu au dependenţe multivaloare ;
   altfel relaţia este în FN4 ;
3. se repetă 1 – 2 pentru fiecare relaţie a structurii ;

Definiţie :

„O relaţie R(A,B,C) menţine o dependenţă multivaloare A →→ B dacă şi numai dacă mulţimea valorilor lui B care corespund unei perechi date ⟨*valoare* A, *valoare* C⟩ din R depinde numai de ⟨*valoare* A⟩ şi este independentă de valorile lui C (A →→ B/C)".

FN5 :: transpunerea schemei relaţionale în FN5 :

1. se ia o relaţie şi **se determină dependenţele joncţiune** ;
2. **dacă** relaţia posedă dependenţe joncţiune **atunci** relaţia se proiectează în relaţii care nu posedă dependenţe joncţiune ;
   altfel relaţia este în FN5 ;
3. se repetă 1–2 pentru fiecare relaţie a structurii ;

Definiţie :

„O relaţie R este în FN5 dacă şi numai dacă fiecare dependenţă joncţiune este implicată printr-un candidat cheie al lui R".

Notă :

Pentru aprofundarea noţiunilor şi a modului de determinare practică a acestora se poate consulta, de către cei care doresc acest lucru, cursul „Baze de date relaţionale", litografiat A.S.E.-Bucureşti, 1985.

3) Transpunerea schemei relaţionale obţinută la pasul 2 într-o schemă conceptuală, reprezentată grafic, acceptată de un alt tip de SGBD şi descrierea schemei cu ajutorul LDD.

Acest pas este legat strict de un anumit SGBD. Transpunerea schemei se va realiza ţinînd cont de specificaţiile de descriere a structurilor bazelor de date gestionate de acest tip de SGBD. Această transpunere trebuie să conserve optimizările obţinute prin normalizare. Pentru structurile gestionate cu SGBD-SOCRATE trebuie să se ţină cont, la realizarea proiecţiilor, de faptul că fiecare entitate de tip SOCRATE posedă un număr maxim de realizări care trebuie recalculat la executarea proiecţiei. Acest număr maxim de realizări nu trebuie, neapărat, să fie conservat ca valoare, în toate proiecţiile deoarece prin proiecţie se asigură conservarea informaţiilor necesare aplicaţiilor funcţionale şi o reducere a redundanţei datelor la nivel de tuplu (realizare). La transpunerea schemei se va analiza dacă o cheie compusă trebuie conservată prin structurile iniţiale ale atributelor (care o compun) sau se va transforma în cîmpuri în care se vor stoca numere de realizări. Cheile, în întregime sau numai pe porţiuni, pot fi întocmite sau dublate cu relaţii de tip ⟨*inel*⟩ ⟨*referire*⟩ sau cu caracteristici ⟨*inverse*⟩.

La alegerea între o legătură de tip ⟨*inel*⟩-⟨*referire*⟩ şi o caracteristică de tip ⟨*invers*⟩ se va ţine cont de spaţiul de memorie ocupat de acestea, de modul de acces la realizările la care se referă (⟨*invers*⟩-acces direct ; ⟨*inel*⟩ acces secvenţial) şi de faptul că o caracteristică de tip ⟨*inel*⟩-⟨*referire*⟩ este gestionată în mod automat de SGBD iar una de tip ⟨*invers*⟩ trebuie gestionată de programele utilizatorului. Această gestiune a utilizatorului poate fi uşurată prin scrierea unor primitive de acces la caracteristicile de tip ⟨*invers*⟩ (sub formă de macroinstrucţiuni), primitive utilizate în toate programele care execută operaţii asupra acestora. Mai mult, definirea identificatorilor caracteristicilor poate fi subordonată unor reguli formale care permit construirea prin investiţii minime de programare, unor generatoare automate capabile să realizeze codul sursă al tuturor primitivelor de acces la acest tip de caracteristici. Utilizarea unor tehnici care să permită generarea automată a codului sursă al unor programe, care realizează diverse operaţii cu entităţile din baza de date (creare, actualizare, modificare, ştergere, listare, testare a coerenţei logice, citare parţială sau totală etc.) aduce un plus de optimizare a realizării bazei de date. O bază de date este realizată în momentul în care este înzestrată cu pro-

gramele de întreţinere a datelor pe care le gestionează. Modul de formalizare a identificatorilor în vederea generării automate a programelor a fost definit în [ACS82] şi [IACS83].

La realizarea descrierii structurii cu ajutorul LDD-SOCRATE se va alege între construirea unei singure baze de date sau construirea mai multor baze de date, divizate funcţional, care să permită montarea singulară şi montarea în paralel în vederea exploatării simultane cu ajutorul interfeţei cu limbaje evoluate [AS84].

4) Optimizarea structurii din punct de vedere al spaţiului fizic ocupat şi al timpului de acces la informaţii.

Ca şi precedentul pas, acest pas, este legat în mod strict de un anumit SGBD. Deşi o bază de date presupune o detaşare totală a programatorilor de aplicaţie, şi în general a utilizatorilor (în afara programatorilor de sistem) de modul de stocare a datelor şi modul de acces pe suportul extern, cunoaşterea reprezentării fizice şi a modului de acces permite optimizarea structurii bazei de date.

Este evident faptul că această cunoaştere nu se referă la întreaga gamă a utilizatorilor ci la un utilizator particular al bazei de date — administratorul — al cărui rol este de a defini structura bazei de date, modul de acces la date, drepturile de acces ale utilizatorilor etc.

În continuare vom prezenta modul de optimizare a structurii pentru SGBD SOCRATE, insistînd asupra aspectelor optimizării specifice acestuia.

Optimizarea structurilor şi a programelor pentru baze de date gestionate cu SGBD-SOCRATE

Un obiectiv major urmărit în maximizarea eficienţei exploatării unui sistem informatic (implicit, deci, în maximizarea eficienţei globale a sistemului informatic) este acela al realizării unui „compromis optim" între resursele de memorie (internă şi externă) utilizate şi timpul mediu de acces necesar regăsirii unei informaţii stocate pe suportul extern. În conformitate cu cerinţele acestui obiectiv este necesar să se acorde o atenţie deosebită optimizării structurii bazei de date, optimizare care să urmărească minimizarea resurselor de memorie (internă şi externă) utilizate şi a timpului mediu de acces la informaţiile stocate în memoria externă.

Realizarea acestui obiectiv — privit din punct de vedere al descrierii datelor — presupune o dublă optimizare a structurilor descrise cu LDD, şi anume :

— optimizarea structurilor la nivelul spaţiului virtual ;
— optimizarea structurilor la nivelul spaţiului real.

Pentru a trata această problemă vom face mai întîi o scurtă prezentare a celor două spaţii de memorie care ne interesează.

Niveluri de organizare şi gestiune a memoriei

Aşa cum am arătat în capitolele anterioare realizarea unei baze de date presupune evidenţierea a trei niveluri :

— nivelul extern ;
— nivelul conceptual ;
— nivelul intern.

Vom face o analiză mai aprofundată a modului de tratare, în cazul nostru, a nivelului conceptual şi a nivelului intern.

Nivelul conceptual este reprezentat, de obicei:

— sub formă de schemă conceptuală descrisă cu ajutorul unui model de reprezentare grafică. Această formă este utilizată pentru a oferi o imagine sintetică, de ansamblu, a schemei conceptuale a bazei de date;

— sub formă de declaraţii efectuate în LDD al SGBD ales. Această formă este aceea care va fi acceptată şi gestionată efectiv de către calculator.

Considerînd baza de date un „fişier" această formă se comportă ca o „structură logică" a datelor pe care le va gestiona. Extrapolînd, putem considera acest nivel conceptual, descris în LDD, ca **„nivel logic"** al bazei de date. Orice program de manipulare a datelor din baza de date, scris în LMD, se va raporta în mod automat la acest nivel.

După compilare acest „nivel-logic", care reprezintă elementul de referinţă în realizarea operaţiilor cu datele stocate în baza de date, va forma *memoria virtuală* a bazei de date. Imaginea acestei memorii virtuale este asociată fiecărei baze definite şi este stocată în spaţiul disc asociat acesteia. Această memorie virtuală reprezintă spaţiul virtual al bazei de date.

Nivelul fizic: reprezintă spaţiul fizic ocupat efectiv de datele asociate unei structuri logice sau unei părţi a unei structuri logice. Pentru o bază de date, spaţiul fizic, asociat, poartă denumirea de spaţiu real.

Problema principală a gestiunii memoriei, în spaţiul virtual sau real, este aceea a alegerii modului de alocare a spaţiilor de memorie:

— *dinamic* — fiecărei informaţii i se alocă spaţiul cerut la momentul introducerii în sistem la o adresă oarecare (cu pierderea unui spaţiu important de memorie necesar gestiunii). Această metodă se impune în toate cazurile în care prevederea dimensiunii memoriei maxime pe care o alocăm unei informaţii nu poate fi făcută înaintea execuţiei;

— *static* — fiecare informaţie care poate să apară ocupă un spaţiu de mărime prestabilită la o adresă determinată. Această metodă este aplicabilă dacă şi numai dacă se poate prevedea înaintea execuţiei mărimea maximă a informaţiei supuse prelucrării.

SOCRATE utilizează pentru gestiunea spaţiilor de memorie tehnica statică căreia îi îmbunătăţeşte performanţele de utilizare prin definirea unei funcţii de transfer realizată logic, între spaţiile cu care lucrează: virtual şi real. În SOCRATE/II această funcţie este cablată reprezentînd astfel un precursor al unei „maşini de baze de date" SOCRATE.

Organizarea spaţiului virtual şi a spaţiului real

Principiul care stă la baza tehnicii statice este acela al minimizării numărului de accese, pe suportul extern, necesar regăsirii unei informaţii. Pentru realizarea acestui deziderat este necesar să existe posibilitatea determinării adresei (pe suportul extern) fiecărei informaţii stocate în baza de date.

Pentru aceasta se defineşte o memorie virtuală, suficient de mare, în care fiecare informaţie poate exista iniţial la o adresă bine determinată. Această adresă va reprezenta elementul de bază utilizat pentru memorarea şi regăsirea informaţiei pe suportul extern.

Unitatea elementară adresabilă pentru spaţiul virtual sau spaţiul real este bitul. Biţii vor fi regrupaţi în cuvinte, dimensiunea unui cuvînt fiind dependentă de implementare (caracteristică a calculatorului gazdă).

Adresa virtuală, a oricărei informaţii, va fi codificată pe un cuvînt.

Pentru explicaţiile ulterioare vom considera dimensiunea cuvîntului de memorie de 32 biţi.

În acest caz mărimea memoriei virtuale nu trebuie să depăşească $2^{31} - 1$ cuvinte.

În această memorie virtuală fiecărei informaţii i se asociază o adresă formală ∂f, definită la momentul compilării :

$\langle \partial f \rangle ::= (\langle Q \rangle \langle M \rangle)$

$\langle Q \rangle$: adresa primei apariţii a informaţiei ;

$\langle M \rangle$: distanţa între două apariţii succesive a aceleiaşi informaţii (distanţa luată în sensul definiţiei formale) iar fiecărei clase de entitate i se asociază numărul maxim de realizări posibile (Nmre). Exemplu :

Considerăm entitatea JUDEŢE cu maxim 64 de realizări care are structura prezentată în fig. 8.2.

Dacă analizăm două realizări ale entităţii şi considerăm :

— Q1 : adresa virtuală de origine a primei realizări a atributului JUD-LOC ;

— M : dimensiunea unei realizări de entitate ;

— Q2 : adresa virtuală de origine a primei realizări a atributului DEN ;

atunci, adresa atributelor JUD-LOC şi DEN din cea de-a doua realizare va fi (Q1, M) şi respectiv (Q2, M).

Realizarea corespondenţei între memoria virtuală (Mv) şi memoria reală (Mr) presupune o translaţie a Mv în Mr (sau Mr în Mv).

Memoria virtuală definită este de fapt o memorie foarte mare în care pot exista toate informaţiile definite în structura BD. În realitate nu toate obiectele sau fenomenele reale, din clasa celor supuse analizei, sînt caracterizate de aceste informaţii considerate în totalitate. Apare necesar deci ca memorarea informaţiilor, pe suportul extern, să se facă numai pentru acelea care există în realitate.

De exemplu, dacă domeniul pe care îl modelăm este activitatea de personal vom defini un atribut care să specifice dacă persoanele căsătorite lucrează în aceeaşi unitate. Acest atribut caracterizează domeniul dar pentru majoritatea cazurilor este posibil să fie vid (cel puţin pentru persoanele necăsătorite).

Realizarea corespondenţei între cele două tipuri de memorie (corespondenţa biunivocă) presupune existenţa unui mod unic de organizare a lor. Modul de organizare ales este următorul :

— spaţiul virtual şi cel real este paginat (dimensiunea unei pagini este un număr întreg de cuvinte, de exemplu 1 ko, pentru V 1.5 şi V 1.6.R) ;

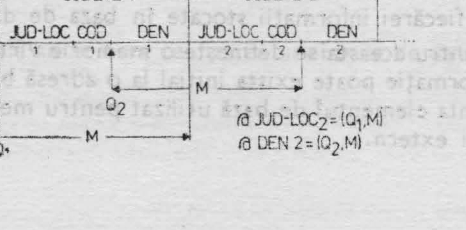


Fig. 8.2. Principiul asocierii adreselor caracteristicilor care aparţin unei realizări de entitate

— fiecare pagină este împărţită în subpagini ;

— fiecare subpagină conţine un număr întreg de cuvinte, dimensiunea ei reprezentînd un parametru ales de utilizator, la definirea bazei de date, în funcţie de proprietăţile datelor supuse analizei ;

— numărul de subpagini conţinute de o pagină se determină în funcţie de dimensiunea subpaginii. În V 1.5 şi V 1.6.R este necesar ca dimensiunea aleasă pentru subpagină să permită divizarea completă a paginii, adică să fie un submultiplu al lui 256.

Pentru înscrierea unei informaţii, în baza de date, facem următoarele precizări asupra modului de utilizare a subpaginilor :

— orice caracteristică poate să ocupe mai multe subpagini ;

— o subpagină poate conţine mai multe caracteristici ;

— ocuparea subpaginii se face în bloc : atunci cînd o dată elementară există se înregistrează subpagina virtuală corespunzătoare, în memoria reală (suportul extern), iar locul ocupat, de această subpagină, este rezervat acestei date şi eventual celorlalte date elementare conţinute în subpagina virtuală. Condiţia ca o subpagină din spaţiul virtual să fie proiectată în spaţiul real este ca subpagina virtuală să conţină cel puţin un bit de informaţie.

Corespondenţa dintre adresa virtuală a unei informaţii şi adresa sa reală se realizează prin aplicarea unei funcţii de dispersie a („hashing") adresei virtuale. Această funcţie trebuie să aibă două calităţi :

— să fie rapidă (în execuţie) ;

— să minimizeze coliziunile (o coliziune apare în momentul în care prin aplicarea funcţiei de dispersare se obţine aceeaşi adresă de pagină pentru valori diferite ale adresei virtuale iar pagina determinată este ocupată).

Pentru explicarea modului de funcţionare a acestei funcţii de dispersie, convenim următoarele :

— volumul memoriei reale este V_r ;

— volumul memoriei virtuale este V_v ;

— mărimea unei pagini este de 2^p biţi ;

— numărul paginilor reale este 2^r ;

— numărul paginilor virtuale este 2^v ;

— adresa reală a informaţiei i este ∂r_i.

În aceste condiţii funcţia de dispersare face să corespundă adresei reale a unei informaţii adresa sa virtuală luată (modulo V_r) :

$$\partial r_2 = \partial v_2 \text{ (modulo } V_r)$$

Această funcţie permite, plecînd de la adresa virtuală, să se determine :

— adresa paginii (*ap*) ;

— adresa subpaginii în pagină (*asp*) ;

— adresa octetului în subpagină (*k*).

Fie *x* o adresă virtuală de $(v + p)$ cifre binare. Structura formală a acestei adrese şi modul de descompunere în componente este :

⟨*x*⟩ : : = ⟨*avp*⟩ ⟨*al*⟩

⟨*avp*⟩ : : = ⟨*ind*⟩ ⟨*ap*⟩

⟨*al*⟩ : : = ⟨*asp*⟩ ⟨*k*⟩

⟨*avp*⟩ : adresa virtuală a paginii ;

⟨*al*⟩ : adresa liniei în pagină.

⟨*ind*⟩ : partea de adresă virtuală care se conservă în vederea recompunerii adresei virtuale (pentru găsirea informaţiei din structură, corespunzătoare unei informaţii reale Mr → Mv).

Acest element precizează indicativul subpaginii virtuale a cărei lungime este :
⟨*ap*⟩ : adresa paginii ;
⟨*asp*⟩ : adresa subpaginii în pagină ;
⟨*k*⟩ : adresa octetului în subpagină ;
⟨*x*⟩ : : = ⟨*ind*⟩ ⟨*ap*⟩ ⟨*asp*⟩ ⟨*k*⟩/⟨*ind*⟩ ⟨*ap*⟩ ⟨*al*⟩

Pentru fiecare subpagină reală se rezervă, în cadrul paginii din care aparține, un cuvînt de control (*cc*) conform modului ilustrat în figura 8.3.

Cuvintele de control asociate subpaginilor din cadrul unei pagini sînt grupate la începutul paginii și au rolul de a permite :

– regăsirea adresei subpaginii virtuale corespondente ;
– evitarea coliziunilor, pentru a optimiza căutarea pe suportul extern ;
– recuperarea subpaginilor devenite libere.

Structura cuvîntului de control ⟨*cc*⟩ este :

⟨*cc*⟩ : : = ⟨*ind*⟩ ⟨*ptr*⟩ ⟨S⟩ ⟨SQ⟩ ⟨NI⟩

⟨*ind*⟩ : partea de adresă virtuală care este conservată pentru a putea distinge, în cazul coliziunilor, cărei adrese virtuale i se asociază ;
⟨*ptr*⟩ : punctator care înlănțuie subpaginile virtuale care fac coliziune (două subpagini fac coliziune atunci cînd adresa lor virtuală nu diferă decît prin ⟨*ind*⟩) ;
⟨S⟩ : semnul lui ⟨ptr⟩ (⟨*ptr*⟩ este algebric și relativ) ;
⟨SQ⟩ : bit care reprezintă dacă subpagina este un „squatter" (SQ = 1) sau nu (SQ = 0).

Definiție :

O subpagină reprezintă un „squatter" dacă locul său în spațiul real a fost obținut prin rezolvarea a cel puțin o coliziune ;

⟨NI⟩ : numărul de octeți ocupați în subpagină.

Cu ajutorul acestor cuvinte de control se asigură înlănțuirea în inel a subpaginilor a căror adresă virtuală nu diferă decît prin ⟨*ind*⟩. Această înlănțuire facilitează procesul de regăsire a unei subpagini din spațiul real (plecînd de la subpagina virtuală) sau a unei subpagini din spațiul virtual (plecînd de la subpagina reală).

Practic, cele două sensuri de regăsire, permit să se asocieze :

– datele reale unei structuri (real → virtual) ;
– o structură unei date reale (virtual → real).

Asupra modului de înlănțuire a subpaginilor facem următoarele precizări :

– primul element din lanț nu este „squatter" (dacă este, i se schimbă, eventual, locul pînă ce îndeplinește această condiție) ;
– ultimul element din lanț punctează pe primul (realizarea înlănțuirii în inel).

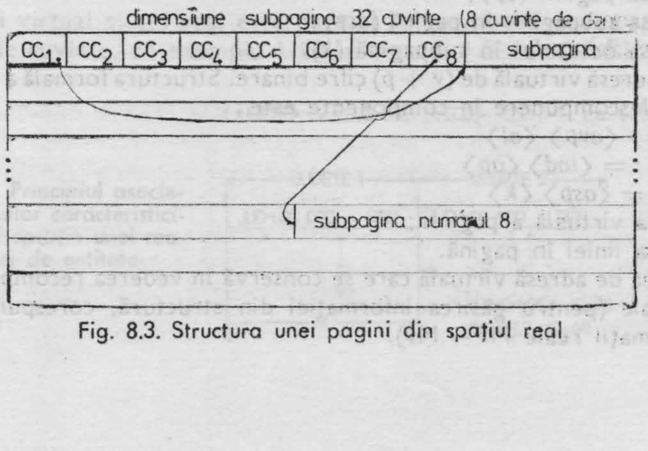


Fig. 8.3. Structura unei pagini din spațiul real

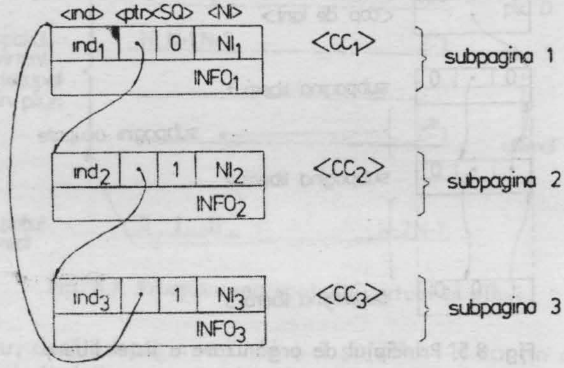


Fig. 8.4. Principiul realizării înlănţuirilor subpaginilor din spaţiul real

Schematic, modul de înlănţuire a subpaginilor, poate fi reprezentat ca în figura 8.4.

$\langle CC_i \rangle$: cuvîntul de control al subpaginii i;

$\langle INFO_i \rangle$: datele conţinute de subpagina i.

Corespondenţa dintre spaţiul real (R) şi spaţiul virtual (V) se realizează astfel :

RV1 : : se stabileşte subpagina reală, caracterizată de $\langle ind_i \rangle$ căreia dorim să-i găsim corespondenţa virtuală ;

RV2 : : **DACA** subpagina reală este prima subpagină din lanţ

ATUNCI sări la RV3

ÎN CAZ CONTRAR se parcurge inelul (prin $\langle ptr \rangle$) pînă se găseşte prima subpagină (SQ = 0) ;

RV3 : : numărul paginii, care conţine această subpagină, ne dă variabila $\langle ap \rangle$ iar numărul subpaginii în pagină ne dă $\langle asp \rangle$, deci $\langle avp \rangle ::= \langle ind_i \rangle \langle ap \rangle$, iar $\langle x \rangle ::= \langle ind_i \rangle \langle ap \rangle \langle asp \rangle \langle k \rangle$ (*k* în acest caz nu este utilizat deoarece dorim să găsim adresa unei subpagini).

Din modul de realizare a corespondenţei spaţiu virtual-spaţiu real rezultă că înlănţuirea precedenţelor este independentă de pagini. Această independenţă presupune crearea unei liste libere, de subpagini vide, care să faciliteze căutarea subpaginii corespondente. Crearea listei libere presupune o **formatare** prealabilă a bazei de date (a spaţiului rezervat în memoria externă) care să realizeze :

— rezervarea unui cuvînt de control, la nivel de pagină, care va constitui $\langle$*capul-de-lanţ*$\rangle$ al listei libere ;

— rezervarea unui cuvînt de control, pentru fiecare subpagină liberă, care să conţină un punctator către următoarea, şi un punctator către precedenta subpagină.

Aceste informaţii sînt suficiente pentru a facilita căutarea unei subpagini şi punerile la zi în lista liberă. Cu acest mod de organizare rezolvarea coliziunilor se reduce la căutarea primei subpagini din lista liberă. În cazul în care pagina este plină (lista liberă este vidă) căutarea se poate extinde în pagina următoare. Modul schematic de organizare şi înlănţuire a subpaginilor din lista liberă este ilustrat în figura 8.5.

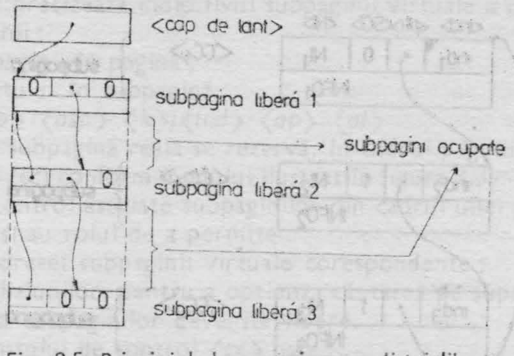


Fig. 8.5. Principiul de organizare a listei libere

Principiul proiecţiei spaţiului virtual (Mv) pe spaţiul real (Mr) şi invers

O condiţie esenţială pe care trebuie să o îndeplinească un sistem de gestiune a bazelor date este aceea de a oferi posibilitatea exploatării, unei/unor baze de date, la modul „multiconsolă-multiprelucrare". "Multiconsolă" în sensul că sistemul trebuie să permită accesul simultan, pentru acelaşi „fişier" (bază de date) de la mai multe console (terminale) şi „multiprelucrare" în sensul că sistemul trebuie să admită apelul mai multor programe de la acelaşi terminal (şi apelul simultan al aceluiaşi program de la mai multe terminale).

Unitatea elementară de transfer între memoria virtuală şi memoria reală este reprezentată de **pagină**.

Proiecţia spaţiului virtual pe spaţiul real este dictată de conţinutul subpaginilor şi se efectuează pe subpagini.

Presupunem că dimensiunea unei subpagini este de 2^8 cuvinte. În acest caz memoria virtuală va fi descompusă în $2^{28} - 1$ subpagini.

Această dimensiune foarte mare pentru majoritatea bazelor de date, face ca spaţiul virtual să fie în general un spaţiu vid. Mărimea spaţiului real asociat unui spaţiu virtual este, în general, mult inferioară acestuia. Dimensiunea va fi aleasă de utilizator în funcţie de mărimea şi natura datelor care se vor stoca efectiv (non nule). Considerăm că spaţiul real are dimensiunea de N subpagini.

În acest context cele două spaţii pot fi reprezentate grafic ca în figura 8.6.

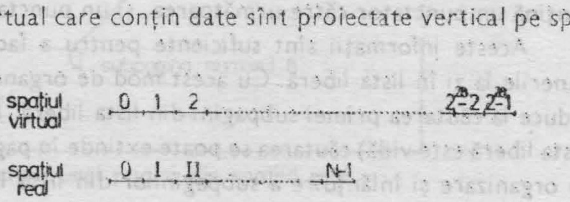
Principial proiecţia spaţiului virtual pe spaţiul real se realizează astfel:

1) se fracţionează spaţiul virtual în „pliuri".

Fiecare pliu va conţine N subpagini (dimensiunea spaţiului real). Pliurile sînt plasate „unele după altele" şi „deasupra spaţiului real" ca în figura 8.7.

2) subpaginile spaţiului virtual care conţin date sînt proiectate vertical pe spaţiul real.

Fig. 8.6. Reprezentarea grafică a unui spaţiu virtual şi a unui spaţiu real descompuse în subpagini



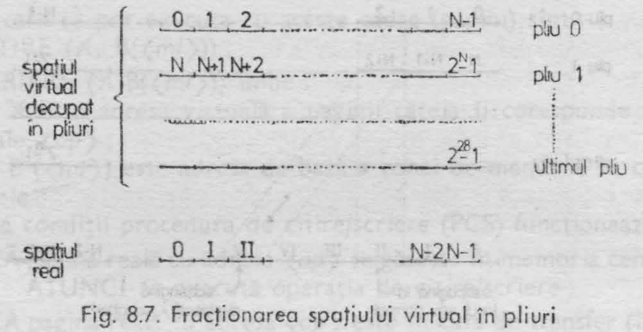


Fig. 8.7. Fracţionarea spaţiului virtual în pliuri

De exemplu, dacă subpaginile 2, N + 1 şi ($2^{28} - 1$) conţin date, atunci vor fi scrise în subpaginile II, I şi respectiv N − 1 din spaţiul real (figura 8.8) conform adreselor obţinute aplicînd funcţia de dispersare : $\partial r_i = \partial v_i$ (modulo N).

3) din modul de construire a funcţiei de dispersare constatăm că există mai multe subpagini din spaţiul virtual care au drept corespondent o anumită subpagină din spaţiul real. De exemplu, dacă dorim să proiectăm subpagina (N + 2) acesteia îi corespunde subpagina II din spaţiul real care a fost ocupată. Această stare aparentă este ceea ce am denumit în paragraful anterior „coliziune". Dacă am efectua efectiv proiecţia, informaţia noastră ar fi denaturată. Pentru a evita această denaturare s-a introdus noţiunea de „squatter", care reprezintă o „subpagină deplasată" din spaţiul real, ca rezultat al unei „excepţii de la regula proiecţiei verticale". Această excepţie se traduce prin : dacă locul destinat unei subpagini din spaţiul real este ocupat de *o altă* subpagină virtuală atunci subpagina virtuală va fi proiectată în prima subpagină liberă din spaţiul real.

În figura 8.9 este reprezentat modul de obţinere al unui „squatter", pentru subpagina (N + 2) considerînd că subpagina V este prima subpagină liberă.

Orice acţiune asupra bazei de date este subordonată căutării unui (unor) element(e) ale acesteia. Căutarea se efectuează prin intermediul mecanismului de paginare, care incorporează proiecţia, prin operaţii de citire/scriere.

Privită în contextul realizării corespondenţei virtual-real, declanşarea unei operaţii de citire/scriere în spaţiul real se desfăşoară astfel :

— la emiterea unui ordin de citire se determină adresa paginii cu care dorim să lucrăm (conform celor specificate anterior) ;

— în cadrul paginii se detaliază căutarea pînă la nivelul elementar specificat (subpagină, linie, octet, bit) ;

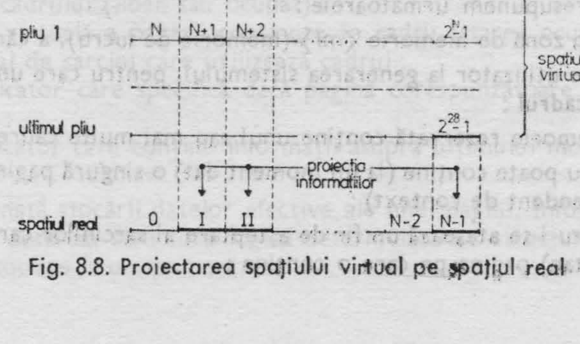


Fig. 8.8. Proiectarea spaţiului virtual pe spaţiul real

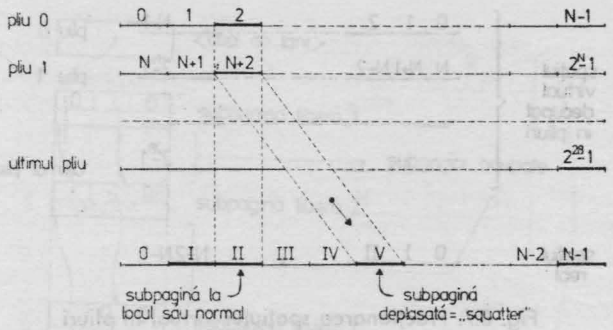


Fig. 8.9. Creearea unui „squatter"

— se efectuează (eventual) modificarea dorită după care pagina trebuie rescrisă în spaţiul real.

Este foarte posibil ca la reluarea unui nou ciclu de operaţii de citire/scriere să dorim să lucrăm cu informaţii din pagina cu care am lucrat anterior (acest lucru este posibil datorită modului de ocupare a subpaginii/paginii de către informaţiile descrise în structură). În aceste condiţii este evident faptul că adoptarea unui mod de lucru de genul citire-modificare elementară-scriere este ineficient.

Evitarea situaţiilor de acest gen se poate face prin păstrarea paginii, o perioadă de timp, în memorie şi rescrierea ei în momentul în care am „terminat" operaţiile pe care dorim să le efectuăm cu datele conţinute de ea.

Să analizăm posibilitatea accesului multiplu (din partea mai multor utilizatori) la aceeaşi pagină. Un acces simultan la aceeaşi pagină poate fi dezastruos (din punct de vedere al conţinutului informaţional) dacă nu stabilim un anumit sistem de priorităţi. Să presupunem că acest sistem de priorităţi este stabilit la modul : fiecare utilizator îşi efectuează toate operaţiile (consecutive) asupra unei pagini, după care o eliberează, pentru un altul. Şi în acest caz apare evident faptul că o pagină, odată adusă în memorie, trebuie să rămînă aici o perioadă de timp, înainte de rescriere. Păstrarea unei pagini în memoria internă presupune ocuparea, o perioadă de timp, a unui spaţiu de memorie operativă. Acest spaţiu poate fi foarte mare dacă ţinem cont de faptul că putem utiliza simultan un număr oarecare de pagini (utilizarea se referă la acelaşi program utilizator în particular, sau la programe diferite — utilizatori diferiţi — în conformitate cu accesul multiplu) şi că sistemele de prelucrare automată a datelor impun (încă ?) restricţii privind dimensiunea zonei de memorie internă utilizată. Din aceste considerente rezultă necesitatea stabilirii modului optim de citire/scriere a unei pagini.

Pentru exemplificarea modului de funcţionare **a procedurii de citire-scriere a unei pagini** presupunem următoarele :

1 : : să rezervă o zonă de memorie ⟨*ml*⟩ (memorie de lucru), a cărei dimensiune se — stabileşte de utilizator la generarea sistemului, pentru care unitatea de organizare este **cadrul** ;

2 : : zona de memorie rezervată conţine unul sau mai multe cadre ;

3 : : fiecare cadru poate conţine (la un moment dat) o singură pagină oricare ar fi ea (este independent de context) ;

4:: fiecărui cadru i se ataşează un fir de aşteptare al sarcinilor care doresc să utilizeze (simultan) pagina pe care o conţine ;

5 :: ordinele care se pot executa cu aceste cadre (pagini) sînt :
5.1 :: CITIRE (X, B(⟨*ml*⟩)) ;
5.2 :: SCRIERE (X,B(⟨*ml*⟩)) unde :
— X este adresa virtuală a paginii căreia îi corespunde adresa paginii reale ⟨*ap*⟩ ;
— B (⟨*ml*⟩) este adresa de bază a zonei de memorie în care se află cadrele.

În aceste condiţii procedura de citire/scriere (PCS) funcţionează astfel :

PCS1. :: DACĂ pagina reală cu adresa ⟨*ap*⟩ se găseşte în memoria centrală (internă)
ATUNCI se execută operaţia de citire/scriere ;

PCS2. :: DACA pagina reală cu adresa ⟨*ap*⟩ este în curs de transfer (în curs de prelucrare)
ATUNCI se execută operaţia de citire/scriere ;

PCS2. :: DACA pagina reală cu adresa ⟨*ap*⟩ este în curs de transfer (în curs de prelucrare)
ATUNCI sarcina este pusă în aşteptare pînă la apariţia unei întreruperi de sfîrşit de transfer (cel puţin pînă la această întrerupere în cazul în care nu există sarcini prioritare care utilizează pagina) ;

PCS3. :: DACA pagina reală cu adresa ⟨*ap*⟩ nu se găseşte în memoria internă
ATUNCI se iniţializează operaţia de citire din spaţiul real astfel :

PCS3.1 :: DACA există un cadru de memorie liber
ATUNCI acest cadru va primi pagina citită ;

PCS3.2 :: DACA nu există un cadru de memorie liber
ATUNCI eliberează cadrul care are cel mai mare timp de neutilizare în perioada staţionării sale în memoria internă astfel :

PCS3.2.1 :: DACĂ pagina din cadru a fost utilizată numai în operaţii de tip citire„ "
ATUNCI rescrie cadrul cu pagina citită ;

PCS3.2.2 :: DACA pagina din cadru a fost utilizată în operaţii de tip „scriere" („citirea" nu contează în acest caz)
ATUNCI pagina este rescrisă pe suportul extern (se activează în prealabil procedura de recuperare a spaţiului real, descrisă ulterior) iar pagina citită îi ia locul.

Deoarece cadrele (de lucru) sînt independente de context (pot primi orice pagină reală) iar o pagină poate fi folosită de mai mulţi utilizatori simultan fiecare cadru este divizat logic în două zone :

1 — zona antet a cadrului care conţine informaţii cu privire la :
— starea cadrului (liber sau ocupat) ;
— adresa virtuală a paginii conţinute în cadru (starea ocupat) ;
— numărul de sarcini care utilizează cadrul ;
— un indicator care specifică dacă pagina corespunzătoare a fost modificată sau nu ;
— un indicator care conţine informaţii asupra ultimului moment de utilizare etc.

2 — zona destinată stocării datelor efective ale unei pagini. Informaţiile conţinute în această zonă au structura logică determinată de structura logică a paginii virtuale asociate la un moment dat.

Remarcăm faptul că numărul de cadre de memorie asociate unui procesor/program reprezintă un parametru furnizat la momentul execuției. Acest număr este adaptat dinamic, în mod automat, la dimensiunea reală a spațiului de memorie destinat execuției procesorului apelat. Alegerea numărului de cadre permite o optimizare a timpilor de execuție ai programului în sensul creșterii proporționale a vitezei de execuție cu mărirea numărului de cadre rezervate. Evident această mărire a numărului de cadre se efectuează în detrimentul ocupării unui spațiu suplimentar de memorie operativă.

În acest context ansamblul zonelor antet asociate cadrelor de lucru, la un moment dat, formează un veritabil dicționar de gestiune a cadrelor.

Operațiile care se pot efectua cu subpaginile din spațiul real sînt următoarele :

a) citirea unei subpagini (⟨ind_i⟩ ⟨ap_i⟩ ⟨asp_i⟩) ;

b) scrierea unei subpagini (⟨ind_i⟩ ⟨ap_i⟩ ⟨asp_i⟩) ;

c) suprimarea unei subpagini (⟨ind_i⟩ ⟨ap_i⟩ ⟨asp_i⟩) ;

* ⟨ap_i⟩ — permite aducerea în memorie (dacă nu există deja) a paginii care conține subpagina pentru care se dictează acțiunea dorită.

a) Citirea unei subpagini (⟨ind_i⟩ ⟨ap_i⟩ ⟨asp_i⟩)

⟨asp_i⟩ — punctează pe subpagina căutată ;

a1 : : DACĂ ⟨ind_i⟩ = ⟨ind⟩/* dacă au același indicativ de diferențiere
  ATUNCI s-a găsit subpagina căutată ;
  ALTFEL

a2 : : efectuează căutarea în subpagina următoare prin ⟨ptr⟩ :

a2.1 : : DACĂ SQ = 1/* squatter ? */
  ATUNCI reia a1 ;

a2.2 : : DACĂ SQ = 0 /* cap de lanț ? *
  ATUNCI subpagina nu există ;

b) Scrierea unei subpagini (⟨ind_i⟩ ⟨ap_i⟩ ⟨asp_i⟩)

⟨ap_i⟩ : pagina ;

⟨asp_i⟩ : plasează subpagina în pagină ;

b1 : : DACĂ SQ = 0 ȘI ⟨ind⟩ = ⟨ind⟩$_1$ /* prima sub-pagină SQ = 0
  ATUNCI subpagina există deja și este cea căutată ;

b2 : : DACĂ SQ = 0 ȘI ⟨ind⟩ ≠ ⟨ind_1⟩
  ATUNCI se parcurge lanțul prin ⟨ptr⟩ :

  b2.1 : : DACĂ SQ = 1 ȘI ⟨ind⟩ = ⟨ind_1⟩
    ATUNCI aceasta este subpagina căutată ;

  b2.2 : : DACĂ SQ = 0 /* s-a parcurs lanțul iar subpagina n-a fostă găsită *
    ATUNCI se caută în lista liberă o subpagină vidă pentru a fi scrisă și se inserează în lista precedentă ;

b3 : : DACĂ SQ = 1
  ATUNCI locul său este ocupat de un „squatter" :

  b3.1 : : se caută o subpagină vidă în lista liberă,

  b3.2 : : se recopiază „squatter"-ul în acest loc, modificînd punctatorii lanțului căruia îi aparține,

  b3.3 : : subpagina se scrie în vechiul loc al „squatter"-ului ;

b4 : : DACĂ adresa subpaginii este aceea a unei subpagini din lista liberă

ATUNCI :

b4.1 : : se scrie subpagina și se actualizează lista liberă (poziționare la valoarea 1 a bitului asociat în șirul de biți de ocupare a subpaginilor) ;

b4.2 : : se modifică punctatorii lanțului căreia îi aparține subpagina găsită ;

b5 : : în subpagina găsită, conform unuia din cazurile b1 – b4, se caută octetul (*k*) în care/începînd cu care dorim să scriem :

b5.1 : : DACĂ valoarea pe care dorim să o scriem este 0 (zero) binar (nedefinit)

ATUNCI se decrementează ⟨NI⟩ cu o valoare egală cu numărul de octeți pe care îi scriem*).

b5.1.1.: : DACĂ ⟨NI⟩ = 0

ATUNCI se antrenează procesul de recuperare a subpaginii reale ;

b5.2 : : DACĂ valoarea pe care o scriem este diferită de 0 (zero) binar (nedefinit)

ATUNCI se incrementează ⟨NI⟩ cu o valoare egală cu lungimea în octeți a informației pe care o scriem*) ;

Procesul de recuperare a subpaginii reale este acela descris la operația de suprimare (ștergere).

Pentru fiecare din cazurile menționate anterior reprezentăm schematic modul de funcționare al algoritmului de scriere plecînd de la situația inițială prezentată în figura 8.10.

b1 : scrierea subpaginilor (*i*, *a*). (*j*, *a*), (*l*, *a*) nu antrenează după sine modificarea punctatorilor deoarece se obține o subpagină care există deja ;

b2 : scrierea subpaginii (*m*, *a*) antrenează o căutare în lista liberă (deoarece aparține lanțului $i \neq j \neq l \neq m$).

Subpagina este luată din lista liberă și necesită o modificare a punctatorilor (fig. 8.11).

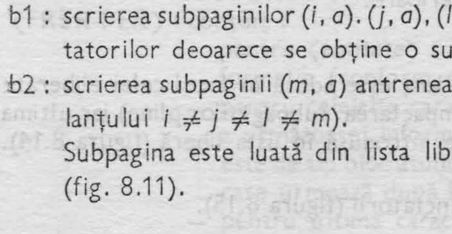


Fig. 8.11. Ocuparea unei subpagini care este „squatter" al listei de subpagini ocupate din lista liberă

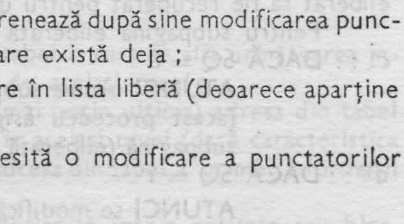


Fig. 8.10. Schema de principiu a unei structuri a spațiului real

---

* În cazul în care lungimea de octeți a valorii pe care o scriem se întinde pe mai mult de o subpagină va fi căutată (și tratată), conform algoritmului descris fiecare subpagină în parte.

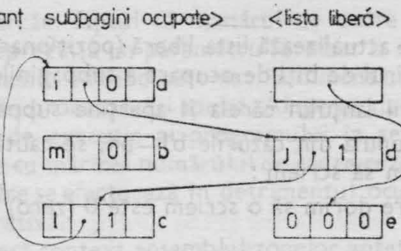


Fig. 8.12. Scrierea unei subpagini al cărei loc este ocupat de un „squatter"

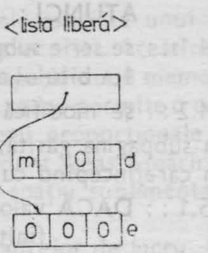


Fig. 8.13. Ocuparea unei subpagini din lista liberă de către o subpagină primară

b3 : scrierea subpaginii (*m*, *b*) : locul pe care trebuie să-l ocupe această subpagină este ocupat de un squatter (*j*, *a*). În acest caz squatter-ul este recopiat într-o subpagină din lista liberă iar subpagina (*m*, *b*) își ocupă locul (fig. 8.12).

b4 : scrierea în subpagina (*m*, *d*) : această subpagină aparține listei libere ceea ce atrage după sine numai modificarea punctatorilor (fig. 8.13).

c) **Suprimarea (ștergerea) unei subpagini** ($\langle ind_i \rangle$ $\langle ap_i \rangle$ $\langle asp_i \rangle$)

Suprimarea unei subpagini are loc atunci cînd toate informațiile pe care le conține sînt șterse prin introducerea caracterului X'00' (nedefinit) (procesul poate apare prin suprimare efectivă (la comanda utilizatorului) sau prin anularea, în timp, valorii informațiilor conținute (contorul de octeți poziționați ⟨NI⟩ ajunge la 0)).

Indiferent de modul în care s-a ajuns la suprimare este necesar ca spațiul real eliberat să fie recuperat pentru utilizări ulterioare.

Pentru subpagina eliberată avem :

c1 : : DACĂ SQ = 0
ATUNCI ultima subpagină din lanț este adusă în locul celei eliberate (acest procedeu asigură recompactarea subpaginilor pline) iar ultima subpagină (eliberată acum) este introdusă în lista liberă (figura 8.14).

c" : : DACĂ SQ = 1
ATUNCI se modifică numai punctatorii (figura 8.15).

Imaginea memoriei virtuale

În acest paragraf vom prezenta modul de obținere și interpretare a imaginii memoriei virtuale pentru versiunea V 1.5. sub sistemele de operare SIRIS 3 sau HELIOS.

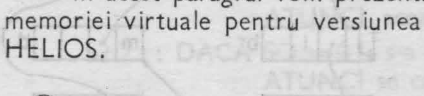


Fig. 8.14. Recuperarea unei subpagini primare

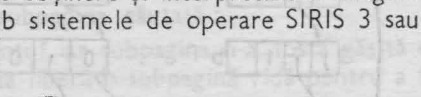


Fig. 8.15. Recuperarea un i „s] at

Această prezentare are scopul de a familiariza cititorul cu modul practic de implementare logică a acestui concept de memorie virtuală. Menţionez că principiile utilizate nu sînt dependente de un anumit tip de calculator sau implementare.

În acest context considerăm că studierea acestui paragraf este oportună indiferent care va fi versiunea pe care cititorul se va decide să lucreze.

În capitolul 3 am considerat faptul că la compilarea structurii, procesorul LDD utilizează tabelele DICOBCK, DICONOM şi BROU pentru a construi structura internă a bazei de date. Această structură internă devine, pentru o bază de date oarecare, dicţionarul bazei de date şi este stocată în spaţiul real afectat acesteia.

Codul intern al structurii poate fi afişat la momentul definirii structurii bazei de date sau la efectuarea unei adăugări (chiar voit eronată pentru a nu altera structura existentă) prin utilizarea comenzilor ACTI TB : 44 şi ACTI TB : 64 înainte de comanda de lansare a compilatorului LDD.

Imaginea obţinută, pe listing sau terminal, sub formă de tabel este construită din linii de forma celor din anexa 1.

Această imagine oferă, în format de reprezentare externă, conţinutul tabelelor DICOBCK, DICONOM şi BROU. Pentru baza de date BDDPERS imaginea memoriei virtuale este prezentată integral în anexa 1.

Structura formală a unei linii este :

⟨ADRESSE⟩ ⟨INDENTIFICATEUR⟩ ⟨FRER-PER⟩ ⟨FILS⟩ ⟨TYP⟩ ⟨CMAX⟩ ⟨ORIGINE⟩ ⟨TAIL PTPR⟩ ⟨ORG⟩ ⟨ADRDIC⟩ ⟨NBMOY⟩

Cîmpurile conţinute de fiecare linie au interpretare diferită în funcţie de tipul de obiect pe care îl descrie, astfel :

⟨ADRESSE⟩ : conţine adresa atribuită de compilator fiecărui identificator definit în structură ;

⟨FRER-PER⟩ : conţine :

- pentru ⟨*entitate*⟩ : adresa din tabel la care se află următoarea informaţie (conform structurii) de acelaşi nivel ;
- pentru celelalte caracteristici (mai puţin ultima) adresa din tabel a următoarei informaţii aflate la acelaşi nivel (dacă caracteristica este de tip bloc atunci conţine adresa din tabel a primei informaţii care urmează după bloc) ;
- pentru ultima caracteristică, declarată într-o entitate sau bloc, conţine o valoare negativă care reprezintă un punctator spre blocul (informaţia) PER asociat ;
- în plus pentru caracteristicile de tip ⟨*invers*⟩ (generate pe 2 linii) conţine în linia a 2-a adresa precedentei informaţii definită în structură la acelaşi nivel ;

⟨FILS⟩ : conţine :

- pentru ⟨*entitate*⟩ : adresa din tabel la care se află blocul său asociat (o ⟨entitate⟩ este un ⟨*bloc*⟩ care are mai multe realizări) ;
- pentru ⟨*bloc*⟩ : adresa din tabel la care se află prima informaţie definită după declaraţia de ⟨*bloc*⟩ ;
- pentru ⟨*referire*⟩ : adresa din tabel la care se află ⟨*entitatea*⟩ la care se face referire ;
- pentru ⟨*inel*⟩ : 0 în cazul în care nu are o caracteristică de tip ⟨*referire*⟩ asociată ;
  - adresa din tabel la care se află caracteristica de tip ⟨*referire*⟩ asociată ;

– pentru ⟨*valoare numerică*⟩ sau ⟨*listă-de-valori*⟩ conţine adresa la care se află dezvoltarea asociată din BROU ;
– pentru ⟨*cuvînt*⟩ sau ⟨*text*⟩ valoarea 0 ;
– pentru ⟨*invers*⟩ :
• linia 1 : adresa entităţii pentru care se ataşează caracteristica ;
• linia 2 : adresa precedentei entităţi definite (înaintea caracteristicii de tip invers) aflată la acelaşi nivel.

⟨TYP⟩ : conţine codul intern utilizat pentru simbolizarea tipului caracteristicii astfel :

2 – entitate	12 – cuvînt formal
3 – invers	15 – cuvînt
6 – inel	16 – listă de valori
7 – referire	17 – text
10 – valoare numerică	19 – bloc
11 – zecimal împachetat	20 – formal
13 – zecimal despachetat	

⟨CMAX⟩ : conţine :
– pentru ⟨*bloc*⟩ valoarea 1 (număr maxim de realizări) ;
– pentru ⟨*entitate*⟩ numărul maxim de realizări posibile declarat ;
– pentru ⟨*text*⟩ numărul de linii ale textului ;
– pentru celelalte caracteristici : dacă caracteristica este declarată cheie de acces cîmpul va conţine numărul de puncte de intrare în spaţiul dicţionar ; în caz contrar conţine valoarea 0.

Pentru caracteristicile de formal conţine :
– pentru ⟨*bloc formal*⟩ valoarea 1 ;
– pentru ⟨*formal logic repetitiv*⟩ numărul maxim de realizări (repetări) ;
– pentru celelalte caracteristici valoarea 0 ;

⟨ORIGINE⟩ : conţine :
– pentru blocul ⟨*FICHIER*⟩ valoarea 65536 ;
– pentru caracteristicile şi entităţile definite la nivel ⟨*FICHIER*⟩ conţine adresa virtuală de origine a acestora (baza adresei este 65536 la care se adaugă spaţiul total alocat caracteristicilor definite anterior) ;
– pentru caracteristicile definite în interiorul unei entităţi (inclusiv entităţi) conţine adresa relativă a cuvîntului de memorie începînd cu care caracteristică îşi ocupă spaţiul alocat (reprezintă distanţa faţă de blocul entitate tată).

Dacă caracteristica este de tip entitate imbricată atunci blocul său asociat va avea ataşată deplasarea sa în entitatea tată iar caracteristicile definite în interiorul său vor avea ataşată deplasarea faţă de blocul său (al entităţii de cel mai mic nivel în care sînt definite) ;

⟨TAIL PTPR⟩ : în funcţie de tipul caracteristicii conţine :
⟨*entitate*⟩ : numărul hexa egal cu mărimea, în cuvinte, a zonei alocate unei realizări de entitate ;
⟨*invers*⟩ : semi-cuvîntul stîng conţine numărul hexa egal cu numărul de biţi alocat contorului asociat şirului de biţi generat ; semicuvîntul drept conţine tipul funcţiei de dispersie utilizate.
⟨*inel*⟩ :: • valoarea 1 pentru lanţ simplu ;
• valoarea 2 pentru lanţ dublu ;

⟨*referire*⟩ : valoarea 1 pentru referire pe nume de entitate ;
- valoarea 2 pentru referire cu lanţ simplu ;
- valoarea 3 pentru referire cu lanţ dublu ;

⟨*cuvînt*⟩ :
- octetul 1 : adresa relativă a începutului caracteristicii în cuvîntul adresă dat de ORIGINE (numărul bitului) ;
- octetul 2 : mărimea caracteristicii exprimată în octeţi ;
- octetul 3 şi 4 : — valoarea X'8000' dacă se utilizează funcţia de H-code pusă la dispoziţie de sistem ;
  — adresa programului de H-code al utilizatorului ;

⟨*valoare numerică*⟩ :
- octeţi 1,3 şi 4 idem cuvînt ;
- octetul 2 mărimea caracteristicii exprimată în biţi ;

⟨*lista-de-valori*⟩ :
- octeţii 1, 3 şi 4 idem cuvînt ;
- octetul 2 arată numărul de valori din listă ;

⟨*bloc*⟩ : idem entitate ;

⟨text⟩ : mărimea de cuvinte ;

⟨*formal*⟩, ⟨cuvînt *formal*⟩, ⟨*valoare numerică*⟩ } : mărimea în octeţi

⟨ORG ADRDIC⟩

⟨*entitate*⟩, ⟨*invers*⟩ } : octetul 1 : numărul de biţi (în hexa) strict necesari pentru reprezentarea cu 2/3/4 octeţi a numărului de cuvinte (hexa) necesare reprezentării lanţului de biţi şi contorului său.

⟨*inel*⟩, ⟨*referire*⟩ } : — octeţii 1—2 neutilizaţi ;
— octeţii 3—4 numărul următoarei linii sursă (hexa) ;

⟨*cuvînt*⟩, ⟨*valoare-numerică*⟩, ⟨*listă-de-valori*⟩ } : — octetul 1 : dacă este diferit de 0 atunci caracteristica este cu acces rapid sau discriminant ;

Semnificaţia biţilor octetului diferit de 0 este :
- bitul 0 : 1 cînd octetul este diferit de 0 ;
- bitul 3 : 1 cheie unică (acces discriminant) ; o cheie dublă (acces rapid) ;
- bitul 4 : 1 lanţ dublu
  0 lanţ simplu
- bitul 5 : 1 lanţ ordonat
  0 lanţ neordonat
- bitul 6 : 1 legătură dublă înainte
  0 legătură dublă înapoi
- bitul 7 : 1 H-code perfect

octeţii 2/3/4 dacă octetul 1 este la 0 are valoarea 0 ; în caz contrar conţine adresa virtuală a începutului dicţionarului (adresa calculată ca multiplu de 512).

⟨NBMOY⟩ : conţine pentru :

— ⟨*referire*⟩ şi ⟨*inel*⟩ : adresa din tabel la care se află inelul, respectiv referirea, asociată ;

— ⟨*valoarea numerică*⟩ : valoarea (în hexa) diferenţei între limitele intervalului declarat ;

— ⟨*lista-de-valori*⟩ :

- semicuvîntul din stînga : numărul de biţi ocupaţi de un element valoare din listă (determinat de numărul de caractere declarate pentru elementul valoare) exprimat în zecimal împachetat ;
- semicuvîntul din dreapta : numărul de elemente declarate în listă ;

— ⟨*entitate*⟩ : punctator spre entitatea înglobată dacă entitatea în curs de prelucrare este referită ;

— ⟨*invers*⟩ : punctator structură pe prima entitate introdusă prin „UN" în citaţia de ⟨*invers*⟩.

Codificarea extensiilor caracteristicilor în BROU se face astfel :

- pentru ⟨*valoare numerică*⟩ se memorează :
  - — marginea inferioară a intervalului de variaţie ;
  - — marginea superioară a intervalului de variaţie ;
  - — marginea superioară extinsă a intervalului de variaţie ;
  - — punctatorul spre identificatorul caracteristicii din DICONOM ;
- pentru ⟨*listă-de-valori*⟩ :
  - — lungimea reală a valorii din listă ;
  - — valoarea definită ;
- pentru fiecare valoare din listă ;
  - — punctatorul spre identificatorul caracteristicii din DICONOM.

Completarea tabelei DICONOM se face cu dublete de forma ⟨adresă⟩ ⟨*identificator*⟩ şi pe principiul unicităţii informaţiei (dacă doi identificatori, descrişi la niveluri diferite (sau locuri diferite) ale structurii, sînt identici vom avea o singură realizare în DICONOM).

Pentru această structură compilatorul LDD verifică unicitatea definirii identificatorilor aflaţi la acelaşi nivel în cadrul blocului (inclusiv bloc ⟨*FICHIER*⟩), corectitudinea sintactică a declaraţiilor de tip, corectitudinea sintactică şi contextuală a citaţiilor de referinţă etc.

Pentru fiecare caracteristică compilatorul determină adresa virtuală a realizării fizice a caracteristicii.

Pentru entităţile aflate în structură la nivel ⟨FICHIER⟩ calculează lungimea în cuvinte a acestora.

Dacă structura compilată este corectă atunci codul intern rezultat este memorat în baza de date pentru a constitui modelul de descriere a datelor „fişierului".

Pe baza adreselor virtuale obţinute (prin utilizarea numărului maxim de realizări posibile ale entităţii avem practic adresa virtuală a realizării fizice a fiecărei caracteristici) prin utilizarea unei funcţii de dispersare se realizează corespondenţa, la momentul exploatării „fişierului" (crearea şi utilizarea „fişierului"), între adresa virtuală a caracteristicii şi adresa sa reală (şi reciproc).

Optimizarea structurilor la nivelul spaţiului virtual

Metoda, utilizată de SGBD-SOCRATE, este aceea de a defini o memorie vizuală, suficient de mare, în care orice informaţie (din structură) poate exista iniţial la o adresă. Fiecare informaţie elementară (caracteristică) ocupă un anumit spaţiu de memorie şi cere un anumit tip de adresă. Introducerea caracteristicilor în această memorie virtuală se efectuează în ordinea secvenţială a definirii lor în structură şi în conformitate cu valoarea la care află contorul de amplasament.

Orice incident apărut la detectarea unei divergenţe între tipul de adresă reprezentat de valoarea contorului de amplasament şi tipul de adresă cerut de caracteristica prezentată la intrare se rezolvă prin adăugarea unei informaţii de cadraj. Aceste informaţii de cadraj reprezintă *„puncte albe"* în structură, în sensul că ocupă întotdeauna din spaţiul rezervat, dar nu sînt adresabile prin metodele puse la dispoziţie de SGBD-SOCRATE.

Optimizarea spaţiului virtual constă, în esenţă, în determinarea acelei ordini a informaţiilor (caracteristicilor), din structură, care duce la minimalizarea spaţiului virtual ocupat.

Această ordine se obţine prin :

1) gruparea caracteristicilor multiplu de cuvînt sau cuvînt (⟨*text*⟩, ⟨*referinţă*⟩, ⟨*inele*⟩, ⟨*invers*⟩) imediat după *DEBUT* (pentru ⟨*entitate*⟩ sau ⟨*bloc*⟩) ;

2) gruparea caracteristicilor de tip ⟨*cuvînt*⟩, ⟨*listă-de-valori*⟩, ⟨*valoare numerică*⟩, de maniera micşorării numărului de biţi neutilizaţi într-un cuvînt memorie.

Optimizarea structurilor la nivelul spaţiului real

Spaţiul real (memoria externă) este împărţit în pagini (de 1 k, de exemplu), fiecare pagină fiind divizată într-un număr întreg de subpagini a căror mărime (număr întreg de cuvinte) reprezintă un element ales de proiectant în funcţie de proprietăţile datelor supuse analizei.

Proiecţia spaţiului virtual, pe spaţiul real, se face pe subpagini de (L – 1) cuvinte. L este parametrul furnizat de proiectant, la generarea bazei de date, şi este asociat spaţiului în dicţionarul datelor. Fiecărei subpagini i se ataşează un cuvînt de control iar cuvintele de control, ale subpaginilor care aparţin unei pagini, sînt grupate la începutul paginii.

Proiecţia unei subpagini din spaţiul virtual într-o subpagină din spaţiul real se efectuează atunci cînd cel puţin un bit din interiorul său are valoarea 1.

Optimizarea spaţiului real constă, în esenţă, în micşorarea numărului de subpagini proiectate.

Această optimizare se obţine prin :

– optimizarea structurii virtuale : reducerea *„punctelor albe"* duce implicit la un cîştig de spaţiu real (în conformitate cu cele prezentate anterior) ;

– optimizarea structurii conform metodei frecvenţelor de apariţie ale caracteristicilor.

Optimizarea structurii bazei de date utilizînd metoda frecvenţelor de apariţie a datelor asociate caracteristicilor

În esenţă o structură a unei baze de date reprezintă un model al datelor reale. Obiectele sau fenomenele (caracterizate de date specifice) supuse analizei se înscriu, în general, într-un caz particular al acestui model (anumite caracteristici nu primesc valori). Este evident deci că pentru o realizare particulară a acestui model putem avea un cîştig de spaţiu real în condiţiile în care proiecţia spaţiului virtual pe spaţiul real se face numai pentru acele caracteristici care au primit o valoare. Am arătat anterior că proiecţia spaţiului virtual pe spaţiul real se efectuează atunci cînd cel puţin un bit, din subpagina respectivă, este poziţionat la 1, iar proiecţia se efectuează în bloc, la nivel de subpagină.

Ideea optimizării constă în găsirea unei modalităţi de grupare a caracteristicilor, din structură, care să ducă la o încărcare cît mai bună a subpaginilor proiectate (prin *gruparea trăsăturilor comune ale modelului şi „izolarea grupată" a trăsăturilor specifice*).

Optimizarea structurilor conform metodei frecvenţelor de apariţie a datelor asociate caracteristicilor presupune parcurgerea următorilor paşi:

F1 : : se determină frecvenţa de apariţie Fa, a tuturor caracteristicilor din structură (mai puţin ⟨*entitate*⟩ şi ⟨*bloc*⟩), astfel:

$$Fa = \frac{Nra}{Nmre}, \text{ unde:}$$

— Nmre : numărul maxim de realizări posibile ale entităţii în care este definită informaţia pentru care se calculează frecvenţa de apariţie ;

— Nra : numărul real de apariţii a informaţiei (pentru toate realizările reale care apar la încărcarea iniţială a bazei de date).

Determinarea lui Nra se poate face astfel:

F1.1 : : *prin metoda observărilor statistice directe*: se efectuează observări statistice pe datele reale care reprezintă lotul de încărcare (creare) a bazei de date ;

F1.2 : : *prin metoda „spionajului" statistic al bazei de date*: se încarcă baza de date (cu structura optimizată în conformitate cu regulile de optimizare a spaţiului virtual şi cu cele de optimizare a accesului la informaţii) cu datele reale, care constituie lotul de încărcare a bazei de date, după care se efectuează teste statistice asupra fiecărei caracteristici (eventual se determină direct Fa asociată) din structură (de exemplu utilizînd funcţia D a limbajului de interogare, pentru a determina numărul de realizări ale caracteristicii cu valoarea diferită de „*nedefinit*"). Deşi metoda pare laborioasă este foarte bună în sensul că:

— sistemul proiectat poate funcţiona cu această structură (dar nu în condiţii optime) ;

— permite efectuarea unor teste de acces asupra caracteristicilor şi determinarea celei mai bune metode de acces utilizabile ;

— permite încărcarea bazei de date, cu structura optimizată, luînd ca intrare baza de date cu structura neoptimizată (trecerea de la o bază la alta poate fi realizată şi la nivel fizic cu programe scrise în limbaj de asamblare).

* Frecvenţa de apariţie se determină pentru fiecare caracteristică în parte şi se asociază acesteia.
* Pentru bloc şi entitate, Fa se determină ca medie a frecvenţelor caracteristicilor aflate pe primul nivel (în interiorul lor).

Observaţie :

Dacă determinarea lui Nra nu se poate face pentru numărul maxim de realizări posibile (Nmre) atunci se poate înlocui acest număr cu un număr egal cu numărul de cazuri supuse observării (Ncso)

Determinarea lui Nra se va face, prin una din metodele descrise, în conformitate cu valoarea lui Ncso. Alegerea acestui număr se va face ţinîndu-se cont de faptul că el reprezintă un eşantion al datelor reale, eşantion care trebuie să fie cît mai reprezentativ pentru a nu denatura evoluţia reală a caracteristicii observate. În acest caz, calculul frecvenţelor de apariţie se va face cu formula : $Fa = \frac{Nra}{Ncso}$, unde Nra se determină în contextul lui Ncso.

F2 : : se ordonează caracteristicile în ordinea descrescătoare a frecvenţelor de apariţie (ordonarea nu trebuie să modifice relaţiile structurale introduse prin îmbricare şi grupare). În cazul în care informaţiile aparţin unui grup, repetitiv sau nu, se ia în considerare numai frecvenţa elementului grup : în interiorul grupului caracteristicile se grupează conform regulei enunţate.

F3 : : se stabilesc *clasele de variaţie a frecvenţelor* caracteristicilor astfel :
se determină numele MS_i şi MI_i ca limită superioară, respectiv inferioară, a intervalului (clasei) de variaţie a frecvenţei i cu proprietăţile :
— MS_1 : cea mai mare frecvenţă de apariţie ;
— $MS_{i+1} < MI_i$;
cu $i = 1, 2, \ldots, n$ unde n este numărul de clase de variaţie stabilite.

F4 : : în fiecare clasă de variaţie a frecvenţei astfel stabilită se grupează caracteristicile conform regulilor de optimizare a spaţiului virtual.

Observaţie :

Această aranjare poate genera apariţia unor informaţii de cadraj de tip „*puncte albe*" atît între două caracteristici succesive din cadrul aceleiaşi clase de variaţie a frecvenţelor, cît şi între clase de variaţie a frecvenţelor adiacente.

„*Punctele albe*" pot fi evitate prin înlocuirea spaţiului rezervat lor cu spaţiul rezervat unor caracteristici din structură (*caracteristici definite*) : ideea este aceea de a obţine, în cadrul clasei, o aranjare care să diminueze la maxim caracteristicile nedefinite „puncte albe".

Această aranjare compactă se poate obţine luînd în considerare următoarele :

F4.1 :: se caută în cadrul clasei (căutarea se face de sus în jos, adică de la caracteristica după care apare primul „punct alb" în jos) o caracteristică care îndeplineşte condiţiile de cadraj şi spaţiu cerute (în particular poate să îndeplinească numai condiţia de cadraj, în cazul în care spaţiul ocupat de această caracteristică este mai mic decît cel necesar completării golului).

DACA caracteristica este găsită în clasa respectivă
ATUNCI va fi transferată în locul „punctului alb" detectat ;

F4.2:: DACA nu există în cadrul clasei o caracteristică corespunzătoare (sau caracteristici corespunzătoare, în cazul în care condiţia cadrajului cerut este îndeplinită de mai multe caracteristici a căror sumă de spaţii ocupate este mai mică sau egală cu spaţiul cerut)
ATUNCI se va căuta o caracteristică (caracteristici) corespunzătoare în celelalte clase, căutarea efectuîndu-se în ordinea de definire a claselor.

Observaţie :

Orice transfer de caracteristică presupune recalcularea spaţiului cerut de informaţiile de tip „punct alb" din clasa din care s-a efectuat transferul.

Metoda este valabilă şi pentru rezolvarea incidentelor de cadraj între clase adiacente (rezolvarea acestor incidente se va face după compactarea clasei imediat superioare).

Structura astfel obţinută este optimă atît din punct de vedere al spaţiului virtual cît şi a celui real.

O problemă care apare la această metodă este aceea a determinării dimensiunii intervalelor de variaţie a frecvenţelor. O modalitate de determinare a intervalelor de variaţie este aceea a împărţirii caracteristicilor (ordonate conform lui F2) pe subpagini. Dacă o caracteristică se întinde pe două subpagini (datorită frecvenţei pe care o are şi nu a spaţiului ocupat — dacă ocupă două subpagini este bine să se analizeze dacă dimensiunea aleasă pentru subpagină este optimă) atunci va fi considerată ca făcînd parte din subpagina următoare (deci frecvenţa ei va fi MS_1 a subpaginii următoare), în subpagina curentă înlocuindu-se cu un „punct alb".

După determinarea intervalelor de variaţie se trece la optimizare conform lui F4.

Metoda împărţirii în clase de variaţie a frecvenţelor, combinată cu cea a optimizării spaţiului virtual şi cu cea a compactării structurii duce evident la un cîştig important de spaţiu la încărcare : subpaginile proiectate vor fi, în acest caz, încărcate la maxim şi de asemenea numărul de subpagini proiectate pe realizare a obiectului real se micşorează la nivel global (prin gruparea caracteristicilor cu frecvenţe mici de apariţie, în subpagini virtuale cu şanse mici de proiecţie).

Dacă informaţiile sînt aranjate în structură într-o ordine aleatoare a frecvenţei, probabilitatea de ocupare a unui număr mai mare de subpagini reale creşte în funcţie de gradul de dispersare a caracteristicilor cu frecvenţe mari de apariţie.

Evident metoda este valabilă numai în cazul în care o realizare a entităţii (aflate la cel mai înalt nivel ⟨FICHIER⟩) ocupă cel puţin două subpagini.

În cazul în care informaţia ocupă numai o subpagină cîştigul din spaţiu de memorie externă este nul dar gruparea, în sine, pe frecvenţe de apariţie a datelor aduce un cîştig sesizabil de timp de exploatare.

Observaţie :

Deşi implementările mai recente ale SGBD SOCRATE aduc o facilitate importantă prin alinierea automată a primei realizări de entitate la frontieră de subpagină, indicăm utilizarea acestei metode. Utilizarea alinierii automate se va face pentru a controla corectitudinea calculelor proprii. Această indicaţie este făcută din considerentul că spaţiul de **cadraj introdus** pentru aliniere în **mod automat** este indispensabil pentru programator. Aceleaşi raţionamente care ne-au adus la ideea optimizării ne obligă să facem utilizabil acest spaţiu, şi din păcate, acest lucru se poate realiza, deocamdată, numai manual sau semiautomat.

Algoritmul metodelor frecvenţei de apariţie este utilizabil pentru optimizarea memoriilor paginate, indiferent de implementare.

Optimizarea timpilor de acces necesari regăsirii unei informaţii

Optimizarea timpilor de acces necesari regăsirii unei informaţii se realizează prin diminuarea numărului de accesuri, la suportul extern (memoria externă). Această diminuare a numărului de accesuri se realizează prin :

1) optimizarea structurii bazei de date ;
2) optimizarea programelor de prelucrare.

Optimizarea structurii bazei de date

Optimizarea este privită, în acest caz, din două puncte de vedere :

a) din punctul de vedere expus anterior orice diminuare a numărului de subpagini proiectate duce implicit la micşorarea timpilor de prelucrare în memoria internă şi a timpilor de transfer al informaţiilor în/din memoria internă din/pe suportul extern ;

b) din punct de vedere al optimizării timpilor de acces la subpagină, structura trebuie să îndeplinească condiţiile :

– prima realizare a fiecărei entităţi trebuie să aibă o adresă multiplu de L – 1 subpagină utilă. Acest lucru se realizează fie mărind spaţiul ocupat de caracteristicile care o preced fie introducînd informaţii de cadraj ;

– fiecare entitate să ocupe un număr întreg de subpagini. Acest lucru se realizează, de asemenea, fie mărind spaţiul ocupat de caracteristicile care o compun, fie introducînd informaţii de cadraj cu frecvenţă de apariţie nulă. În ambele cazuri avem o mărire a spaţiului virtual care nu duce la mărirea timpilor de acces ci la diminuarea acestora, diminuare care se datoreşte îmbunătăţirii condiţiilor de funcţionare a componentelor de acces puse la dispoziţie de sistem.

În cazul realizării unei adrese multiplu de subpagină, pentru prima realizare a entităţii, este recomandabil să se facă următoarea analiză :

– dacă entitatea este la cel mai înalt nivel al bazei de date („FICHIER") atunci se vor introduce variabile de cadraj (aceste informaţii vor reprezenta caracteristici fictive utilizabile : se poate mări numărul de elemente Z_1 şi Y_1 care au o singură realizare) ;

– dacă entitatea este imbricată atunci este eficientă utilizarea unor caracteristici, aflate la acelaşi nivel, drept informaţii de cadraj (dacă se introduce o informaţie de cadraj atunci la orice generare a unei realizări a entităţii înglobante şi la generarea primei realizări a entităţii imbricate acest spaţiu va fi transferat sistematic pe suportul extern).

Algoritm de determinare a dimensiunii informaţiei de cadraj

Algoritmul permite determinarea, pentru entităţile unei baze de date, a informaţiei de cadraj necesară pentru :

– alinierea primei realizări la o adresă de subpagină ;

– alinierea lungimii realizării entităţii la multiplu de subpagină.

Pentru prezentarea algoritmului vom folosi notaţiile :

– *variabile globale* :

NOC : număr octeţi/cuvînt memorie ;
NBC : număr de biţi/cuvînt memorie ;
L : lungimea în cuvinte a unei subpagini ;
NRENTI : numărul de entităţi din baza de date ;

– *variabile pentru o entitate dată* :

NMRE : numărul maxim de realizări ;
NUMENTI : numele entităţii ;
LGENTI : lungimea entităţii ;
LGAL : lungimea recalculată a entităţii ;
LSB : lungimea şirului de biţi în cuvinte ;

AVROG : adresa virtuală de origine a primei realizări ;
N : octeţi calculaţi pentru aliniere la adresă multiplu de subpagină a primei realizări ;
NOCENT : numărul de octeţi de aliniere a lungimii unei realizări la adresă multiplu de subpagină ;
NRSP : numărul de subpagini ocupate de o realizare ;

— *variabile de lucru* : I, MODIF, RLSB, RMODIF, RNSP

— ⊕ : variabila care urmează va primi restul împărţirii ;

```
1 : start ;
2 : iniţializări :
      AVORG  : = 65536 ;
      NOC    : = 4 ;
      NBC    : = 32 ;
      I      : = 1 ;
3 : citeşte NUMENTI, NMRE, LGENTI ;
      N : = 0 ;
      /* calcul lungime şir de biţi ⟨LSB⟩ */
      LSB ⊕ RLSB : = (NMRE / NBC) ;
      Dacă RLSB > 0
          atunci
              LSB  : = LSB + 1 ;
      sfd ;
/* calcul octeţi ⟨N⟩ de aliniere entitate la adresă                 */
/* multiplu de subpagină                                             */
MODIF ⊕ RMODIF : = (AVORG + 2 * LSB + 2) / L ;
dacă RMODIF > 0
      atunci
      MODIF : = MODIF –1 ;
sfd ;
MODIF : = (AVORG + 2 * LSB + 2) – (L * MODIF) ;
dacă MODIF > 0
      atunci
      N : = NOC * (L – MODIF) – 1 ;
/* calcul octeţi aliniere lungime entitate ⟨NOCENT⟩ la               */
/* multiplu de subpagină L                                           */
   NRSP ⊕ RNSP : = (LGENTI/L) ;
dacă NRSP = 0
     atunci
      NOCENT : = (L – RNSP) * NOC ;
      NRSP : = 1 ;
    altfel
      dacă RNSP > 0
          atunci
          NOCENT : = (L–RNSP) * NOC ;
          NRSP : = NRSP + 1 ;
      sfd ;
sfd ;
LGAL : = NRSP * L ;
scrie NUMENTI, AVORG, NMRE, LGENTI, NOCENT, LGAL, LSB, N ;
dacă    I < NRENTI
      atunci
      dacă N ≠ 0
          atunci
              AVORG : = AVORG + (N+1) / NOC ;
      sfd ;
```

```
      AVORG : = AVORG + 2 * LSB + (L * NRSP) * NMRE + 2 ;
      I : = I +1 ;
      salt la 3 ;
   sfd
4 : stop.
```

Optimizarea programelor de prelucrare

Transferul paginilor în/din memoria internă din/în memoria externă se realizează prin intermediul unor „buffere", numite cadre de lucru, astfel :

— la cererea de acces la o pagină se caută mai întîi în aceste cadre de lucru :

— DACĂ pagina căutată există
  ATUNCI nu are loc acces la suportul extern ;

— DACĂ pagina căutată nu există
  ATUNCI se aduce această pagină în cadrul de lucru care conţine pagina cu cel mai mare timp de neutilizare în memoria internă (eventual se transferă, în prealabil, pagina veche pe suportul extern sau se recuperează spaţiul disc dacă pagina a devenit vidă) ;

— la găsirea paginii dorite se detaliază căutarea pînă la nivelul specificat în program.

Ideea de optimizare constă în aceea că odată găsită o pagină şi, în cadrul ei, subpagina să se efectueze toate prelucrările cu informaţiile din această subpagină. Pentru argumentare să considerăm următoarele :

— vom numi distanţa de paginare între două citaţii succesive (care aparţin aceleeaşi unităţi de program) numărul de accesuri la paginare care se efectuează între momentele de execuţie a acestor citaţii (citaţie : apel subpagină-prelucrare informaţie din subpagină) ;

— dacă distanţa de paginare este foarte mare este evident că fiecare citaţie va face apel la citirea paginii de pe suportul extern (o distanţă de paginare mare presupune un transfer important de pagini care poate duce la ştergerea paginii considerate de noi din cadrele de lucru) ;

— ideea este de a anula distanţa de paginare, anulare care se obţine grupînd citaţiile care se referă la informaţii care aparţin aceleiaşi subpagini (vezi algoritmul frecvenţelor-grupare caracteristici pe subpagini).

Determinarea optimă a spaţiului dicţionar al unei baze de date

Pentru toate implementările şi versiunile SOCRATE ABD are posibilitatea stabilirii dimensiunii în unităţi fizice alocabile (dependente de tipul suportului bazei de date şi caracteristicile sistemului de operare gazdă) a spaţiului destinat stocării datelor aferente unui spaţiu virtual. În general, prin această dimensionare se stabileşte de fapt numărul de pagini ale unui spaţiu fizic. Dacă pentru spaţiul fizic destinat stocării datelor există şi posibilitatea determinării mărimii subpaginii pentru celelalte spaţii această mărime este prestabilită (pentru a simplifica, pe de o parte, algoritmii de gestiune, iar pe de altă parte, şi pentru a imprima un caracter de „transparenţă" accentuată a intervenţiei acestora în prelucrările curente).

În acest context utilizatorul (ABD în special) are posibilitatea să determine o dimensiune optimă pentru spaţiul dicţionar al bazei de date utilizînd formula :

$$V_{mc} = [L_{sp} * (N_m * K_m + N_p * K_p + N_e * K_e + \sum_{i=1}^{n} N_{ci} + \sum_{j=1}^{m} N_{rj})] * (K_{uo} + 1)$$

unde :

- V_{mc} : volumul de memorie necesar exprimat în cuvinte ;
- L_{sp} : lungimea unei subpagini în cuvinte ;
- N_m : numărul de programe de tip macroinstrucţiune care se vor stoca în baza de date ;
- K_m : coeficient de corecţie a numărului de macroinstrucţiuni ;
- N_p : numărul de programe precompilate care se vor stoca în baza de date ;
- K_p : coeficient de corecţie al numărului de programe precompilate ;
- N_e : numărul de programe în format executabil tip „sistem operare gazdă" stocate în baza de date ;
- K_e : coeficient de corecţie al numărului de programe executabile.

Coeficienţii K_m, K_p şi K_e se stabilesc în funcţie de evoluţia probabilă a numărului elementelor respective pe parcursul exploatării bazei de date. În general, aceşti coeficienţi asigură un spaţiu suplimentar necesar stocării unor elemente de tipul respectiv şi au aceeaşi lege de definire cu aceea a prevederii aplicaţiilor posibile de realizat pe baza de date respectivă.

N_{ci} : numărul maxim de valori pe care le poate lua caracteristica C_i declarată cheie de acces (rapidă sau discriminantă).

Pentru o caracteristică declarată cheie de acces Nc_i se calculează astfel :

- Nc_i = Nmre, dacă caracteristica este declarată într-o entitate de primul nivel ;
- $Nc_i = \prod_{l=1}^{m} Nmre_l$, dacă caracteristica aparţine unei entităţi de nivel l (m ⩽ Nmi), unde prin $\prod$ am notat operaţia de produs iar prin Nmi numărul maxim de imbricări admise, pentru entitate, de implementare ;
- Nmre : numărul maxim de realizări declarate pentru entitate ;

Nr_j : reprezintă numărul maxim de asocieri de tip referire pentru caracteristica de tip referire j.

Ca şi la declaraţiile de chei de acces Nr_j se calculează astfel :

- Nr_j = Nmre, dacă caracteristica este declarată într-o entitate de primul nivel ;
- $Nr_j = \prod_{e=1}^{m} Nmre_e$, dacă caracteristica este declarată într-o entitate imbricată de nivel m (m ⩽ Nmi).

În cazul în care implementarea SOCRATE pe care o utilizăm nu admite programe direct executabile sau caracteristicile de tip referire nu se stochează în spaţiul dicţionar, elementelor respective din formulă (Ne şi Nr_j) li se acordă valoarea 0.

Ku0 : coeficient de umplere maxim admis pentru subpagină la încărcarea iniţială a bazei de date. Acest coeficient este valabil numai pentru implementările care stochează mai multe elemente de acelaşi tip pe subpagină.

În cazul în care se stochează un singur element pe o subpagină acest coeficient are valoarea zero.

Dacă Noc este numărul de octeţi pe un cuvînt de memorie atunci dimensiunea în octeţi a volumului de memorie (Vmo) necesar este :

$$Vmo = Noc * Vmc$$

Considerînd că Vua este dimensiunea în octeţi a unei unităţi alocabile (în sens sistem de operare gazdă) atunci necesarul de unităţi alocabile (Nua) pentru spaţiul dicţionar se determină astfel :

$$\frac{Vmo}{Vua} = Q + r, \text{ dacă } \begin{cases} r = \text{zero atunci } Nua = Q * Ko \\ r \neq \text{zero atunci } Nua = (Q + 1) * Ko \end{cases}$$

unde Ko reprezintă coeficientul de umplere optim admis pentru un fişier oarecare de către sistemul de operare gazdă.

Dimensiunea spaţiului dicţionar poate fi modificată în orice moment. Această modificare necesită o reorganizare a acestui spaţiu prin operaţiile de restaurare/salvare sens SOCRATE (cap. 6). Indiferent dacă modifică sau nu dimensiunea acestui spaţiu ABD trebuie să analizeze în permanenţă rezultatele aplicării procesorului de statistici pe acest spaţiu. Dacă acest procesor semnalează schimbări frecvente de pagini/subpagini pentru găsirea unei informaţii sau lanţurile de „squatteri" devin prea lungi (ponderea lanţurilor lungi în totalul spaţiului alocat) va efectua o operaţie de salvare/restaurare sens SOCRATE pe acest spaţiu. Operaţia de salvare se va efectua, principial, pe zone de spaţiu *disjunct continue* iar restaurarea pe întreg spaţiu conform principiului descris în capitolul 6.

Tehnici de construire a programelor de serviciu

Controlul coerenţei

Transformările parcurse de o informaţie, descrise în paragrafele anterioare pot fi rezumate astfel :

$$IV \xrightarrow{(f)} Mv \xrightarrow{(g)} Mr \xrightarrow{(h)} Me$$

IV : ansamblul identificatorilor şi valorilor afectate de către utilizator informaţiilor sale prin intermediul LMD ;

Mv : ansamblul adreselor virtuale afectate informaţiilor de către interpretorul LMD utilizînd structura internă a bazei de date ;

Mr : ansamblul adreselor disc (adreselor de pagină şi/sau subpagină) destinate stocării fizice a datelor ;

Mc : ansamblul adreselor din memoria centrală în care sînt prelucrate datele.

Aceste transformări, cu toate că sînt activate prin programele de cereri ale utilizatorului, se execută în mod automat şi sînt transparente acestuia. Funcţiile de transfer (f, g, h) sînt funcţii realizate prin cooperarea unor module software cu diverse componente hardware ale calculatorului.

În mare, operaţiile executate asupra datelor conţinute în baza de date de către un utilizator oarecare pot fi grupate astfel:

1) — operaţii de creare, actualizare, modificare sau ştergere;

2) — operaţii de recuperare şi prezentare.

Aceste două mari grupe de operaţii pot fi combinate.

În cazul operaţiilor de tip 1) utilizatorul trebuie să prevadă toate restricţiile modelului datelor privind integritatea bazei de date (teste de validare, controale de coerenţă logică a unor ansambluri de valori; clasare logică corectă în ansambluri etc.).

În cazul operaţiilor de tip 2) utilizatorul trebuie să modeleze corect algoritmul de regăsire şi prezentare externă conform semanticii logice a acestuia.

Prin funcţiunile programate de utilizator cu ajutorul LMD (sau interfeţei cu limbaje evoluate) care înglobează operaţii de tip 1. utilizatorul poate aduce baza de date (dacă modul de programare a acestora nu respectă restricţiile de integritate ale modelului datelor) cel mult într-o stare de incoerenţă logică, din punct de vedere al utilizatorului, stare care este coerentă din punct de vedere al SGBD SOCRATE. Repararea acestui gen de incoerenţă revine în sarcina utilizatorului deoarece el este acela care cunoaşte semantica datelor iar prelucrarea acestora se efectuează conform modului şi la momentul precizat de el însuşi.

Cu toate că modulele software de realizare a funcţiunilor de transfer permit detectarea erorilor de funcţionare ale componentelor hardware implicate, există anumite cazuri în care o proastă funcţionare a sistemului de calcul poate inhiba însăşi detectarea (inhibarea semnalării erorilor de paritate, a erorilor de transfer etc.). În aceste condiţii sau în cazul producerii unor incidente grave (căderi ale surselor de alimentare, variaţii ale frecvenţei curentului în afara limitelor tolerate etc.) o bază de date poate fi adusă într-o stare de incoerenţă:

— *fizică*: înlănţuiri incorecte ale subpaginilor şi/sau paginilor, neconcordanţe între informaţiile stocate în cuvîntul de control al subpaginilor şi conţinutul acestora etc.);

— *logică*: alterare a conţinutului dicţionarelor asociate caracteristicilor declarate chei de acces („citeşte tabele de indecşi"), alterare a conţinutului şirurilor de biţi sau a caracteristicilor de tip referire cu inel a căror gestiune este în sarcina SGBD SOCRATE.

Starea de incoerenţă fizică este semnalată de către procesorul de statistici şi, în principiu, repararea acestui gen de incoerenţă presupune reluarea prelucrărilor efectuate asupra unei copii a bazei de date definită ca „punct de referinţă" sau întoarcerea, cu ajutorul fişierelor jurnal, la un „punct de control" în care baza de date nu mai prezintă această incoerenţă (cap. 8). O stare de incoerenţă fizică provoacă, în majoritatea cazurilor, în mod automat o stare de incoerenţă logică a datelor.

Detectarea stării de coerenţă logică a acestor elemente gestionate de către sistem dar a căror actualizare este efectuată la cererea expresă a utilizatorului se efectuează prin rularea unor programe cu un algoritm standard. Scrierea acestor programe şi alegerea momentului rulării lor este sarcina administratorului bazei de date care este singura persoană autorizată să le analizeze rezultatul şi să ia o decizie conformă fiecărui caz detectat. Execuţia acestor programe de test este imperios necesară înaintea luării deciziei de construire a unui *„punct de referinţă"* al bazei de date. Modul de construire a acestor programe precum şi cazurile de incoerenţe care se pot detecta vor fi prezentate în paragrafele următoare. Pentru această prezentare vom face referire la schema conceptuală prezentată grafic în figura 8.16.

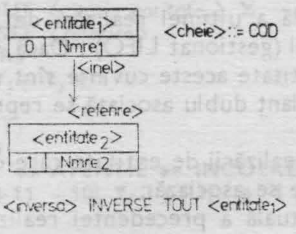


Fig. 8.16. Exemplu de schemă conceptuală

Această schemă poate fi adaptată fiecărui caz particular (chiar dacă declarațiile de tip ⟨inel⟩ și ⟨referire⟩ sînt definite în aceeași entitate) conform declarațiilor efectuate de utilizator. Modificările pe care trebuie să le efectueze acesta se referă la specificarea numelui obiectelor implicate și a citației corecte a acestora.

În general, depanarea incoerențelor detectate se execută prin utilizarea fișierelor jurnal care permit întoarcerea la o stare coerentă. În afara acestei tehnici generale vom sugera unele procedee utilizabile pentru refacerea coerenței în fiecare caz particular.

Caracteristici de tip ⟨*inel*⟩

Pentru a prezenta tipurile de incoerențe care pot fi întîlnite în cazul caracteristicilor de tip ⟨*inel*⟩ vom considera că acestea sînt declarate cu lanț dublu (cele cu lanț simplu sînt definite ca o restricție a acestora).

O caracteristică de tip ⟨*inel*⟩ cu lanț dublu se reprezintă pe 2 cuvinte cu semnificația (figura 8.17) :

— cuvîntul 1 (PRIMUL) : adresa virtuală a primei realizări de entitate care a efectuat referire la acest inel (gestionat FIFO) ;

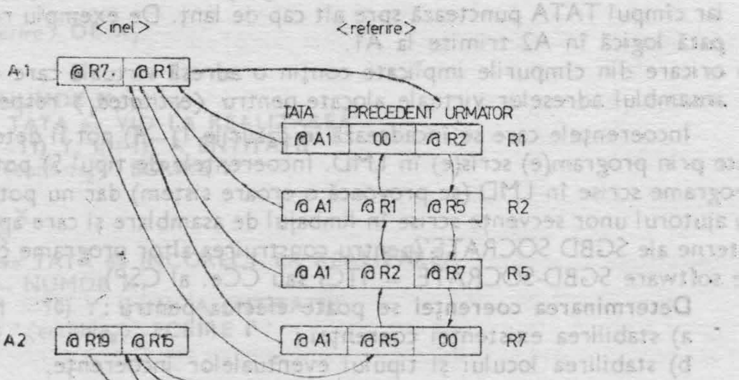


Fig. 8.17. Structura caracteristicilor ⟨ referire cu inel ⟩ și ⟨inel⟩

– cuvîntul 2 (ULTIMUL) : adresa virtuală a ultimei realizări de entitate care a efectuat referire la acest inel (gestionat LIFO) ; Dacă ⟨*inelul*⟩ nu grupează nici-o realizare de entitate aceste cuvinte sînt vide.

Caracteristica de tip ⟨*referire cu inel*⟩ cu lanț dublu asociată se reprezintă pe 3 cuvinte cu semnificația (figura 6.17) :

– cuvîntul 1 (TATA) : adresa virtuală a realizării de entitate care conține caracteristica de tip ⟨inel⟩ la care se asociază ;

– cuvîntul 2 (PRECEDENT) : adresa virtuală a precedentei realizări de entitate care a efectuat referire la același inel (pentru prima realizare din lanț este vid) ;

– cuvîntul 3 (URMĂTOR) : adresa virtuală a următoarei realizări de entitate care a efectuat referire la același inel (pentru ultima realizare din lanț este vid).

Cazurile de incoerență întîlnite la acest tip de caracteristică pot fi :

1) rupturi de lanțuri : pointerul PRECEDENT sau URMĂTOR ajunge la valoarea nedefinit dar mai există realizări de entitate care îndeplinesc această condiție. De exemplu, la realizarea R2 cîmpul URMĂTOR, poziționat pe valoare nedefinită, provoacă excluderea din lanț a realizărilor R5 și R7. Orice parcurgere a realizărilor pornind de la capul de lanț nu permite accesul la aceste realizări grupate logic în același ansamblu.

2) cicluri în lanț : pointerul PRECEDENT/URMĂTOR punctează spre o realizare definită ulterior, respectiv anterior, pe acest lanț. De exemplu cîmpul URMĂTOR al realizării R5 punctează spre realizarea R1. Acest gen de incoerență provoacă o ciclare la orice tentativă de parcurgere a inelului.

3) cîmpul TATA vid : realizarea punctează corect un PRECEDENT și/sau URMĂTOR dar nu trimite la capul de lanț. În cazul unei parcurgeri automate a acestui lanț este imposibilă reîntoarcerea, de la această realizare, la capul de lanț (parcurgeri parțiale ale lanțului imposibile). De exemplu cîmpul TATA al realizării R2 vid.

4) cîmpul TATA incorect : realizarea este grupată într-un lanț (cu un anumit TATA) iar cîmpul TATA punctează spre alt cap de lanț. De exemplu realizarea R15 grupată logică în A2 trimite la A1.

5) oricare din cîmpurile implicate conțin o adresă virtuală care nu face parte din ansamblul adreselor virtuale alocate pentru ⟨*entitatea*$_1$⟩ respectiv ⟨*entitatea*$_2$⟩.

Incoerențele care se încadrează în cazurile 1)–4) pot fi detectate și/sau reparate prin program(e) scris(e) în LMD. Incoerențele de tipul 5) pot fi detectate prin programe scrise în LMD (se provoacă o eroare sistem) dar nu pot fi depanate decît cu ajutorul unor secvențe scrise în limbajul de asamblare și care apelează primitivele interne ale SGBD SOCRATE (pentru construirea altor programe consultați titularul de software SGBD-SOCRATE – ITCI sau CCe. al CSP).

Determinarea coerenței se poate efectua pentru :

a) stabilirea existenței coerenței ;

b) stabilirea locului și tipului eventualelor incoerențe.

a) **Determinarea coerenței globale**

Pentru fiecare inel cu lanț dublu sau simplu se execută următoarea secvență de cereri simple :

M $Y_i = 0$ **M** $Y_j = 0$
POUR TOUT ⟨*citație entitate*$_1$⟩ X_i
  M Y_k=**D TOUT** ⟨*inel*⟩ **DE** X_i
  M $Y_i = Y_i + Y_k$
FIN

```
POUR TOUT ⟨citaţie entitate₂⟩ Xj
    SI EXISTE ⟨referire⟩
      ALORS
        M Yj = Yj + 1
FIN
SI Yi ] = Yj
  ALORS
    I (1) '** ATENTIE ** INCOERENTA LOGICA'
    I (+11 -10) Yi I (+1) '#' I (+11 -10) Yj ECRIRE
FIN
```

Acest mod de detectare permite numai determinarea efectivă a faptului că ansamblul ⟨inel⟩ şi ⟨referire⟩ este incoerent sau nu. În cazul incoerenţelor nu evidenţiază tipul şi locul în care există acestea. Mai mult, funcţia D poate cicla în cazurile detectării unei erori de tipul 2).

b) **Determinarea coerenţei, a locului şi tipului eventualelor incoerenţe**

```
[MYt = ⟨CKp1⟩    ]   /* în cazul în care se doreşte oferirea posibilităţii de */
[DEPUIS ⟨CKp2⟩   ]   /* reluare a programului la incident                     */
                     /* (iniţial ⟨CKp1⟩ şi ⟨CKp2⟩ au valoarea zero)           */

POUR TOUT ⟨citaţie entitatea₁⟩ Xi
M Yj = NUMDE Xi
    [I (1) '* ⟨CKP2⟩ :' I (+11 -10) Yj ECRIRE]
    FAIRE
    M Yi = 0
  POUR TOUT ⟨inel⟩ Xj
   M Yi = Yi + 1
   SI Yi > Nmre₂/* numărul maxim care poate fi grupat                     */
    ALORS M Zi = 'Y'
      I (1) '** CICLARE ** LA REALIZARE'
      I (+11 -10) Yj I(+1) 'A ENTITATII'
      I (+1) '⟨entitate₁⟩' ECRIRE I ''
      SORTIE
    FIN
    M Xk = ⟨referire⟩ DE Xj
    SI PAS Xk
      ALORS
        M Ye = NUMDE Xj
        I (1) '** TATA ** VID LA REALIZAREA'
        I (+11 -11) Ye I(+1) 'A ENTITATII'
        I (+1) '⟨entitate₂⟩' ECRIRE I ''
      SINON
        SI Xk] = Xi
          ALORS
          I (1) '** TATA ** INCORECT LA REALIZAREA '
          M Ye = NUMDE Xj
          I (+11 -10) Ye I(+1) 'A ENTITATII'
          I (+1) '⟨entitate₂⟩' ECRIRE I''
      FIN
    FIN
  FIN
FIN
M Yt = Yt + Yi
FIN
M Xj = U
SI Yt > Nmre₂
  ALORS /* nu pot detecta dacă am cumva şi rupturi, deci :                */
   I (1) '*** GRAV *** CARACTERISTICA'
   I (+1) 'ANALIZATA PREZINTA CICLURI.' I (+1) '!ABANDONEZ!'
```

```
   ECRIRE I' '
   M Y1 = −500
   I (Y1) 'FORTEZ ABANDON SISTEM'
   /* codul de retur al fazei poate fi testat deoarece este mai mare ca 127 */
 SINON
   M Yj = 0
   POUR TOUT ⟨citaţie entitate2⟩Xj
     SI EXISTE ⟨referire⟩
       ALORS
         M Yj = Yj + 1
       FIN
   FIN
   SI Yj < Yt
     ALORS
       I (1) '** RUPTURI ** DE LANTURI'
       I (+1) 'PE REFERIREA' I (+1) '⟨referire⟩'
       I (+1) 'ASOCIATA INELULUI'
       I (+1) '⟨inel⟩' ECRIRE I''
   FIN
 FIN
```

Operaţiile pe care trebuie să le execute utilizatorul în cazul înlăturării incoerenţelor sînt :

— să determine şi să localizeze fiecare tip de eroare ;

— să marcheze ansamblul realizărilor care conţin legături eronate prin cel puţin una din metodele :

• gruparea acestora într-o caracteristică de tip ⟨*invers*⟩ ;

• salvarea elementelor de identificare a realizărilor de tip ⟨*inel*⟩ şi ⟨*referire cu inel*⟩ asociate pe un suport magnetic (pot fi salvate şi în caracteristici definite în acest scop în structura entităţii sau dacă proiectantul nu a prevăzut la realizarea structurii acest lucru în eventualele caracteristici de cadraj) ;

— să anuleze aceste realizări ale caracteristicii ⟨*referire cu inel*⟩ ;

— să testeze, dacă cumva, după această anulare inelul asociat este vid. Dacă inelul nu este vid atunci acesta trebuie anulat fizic pentru a deveni vid ;

— să încarce ansamblul valorilor realizărilor caracteristicii de tip ⟨*referire cu inel*⟩ în baza de date.

În cazul în care trebuie anulat fizic un ⟨*inel*⟩ această anulare se poate efectua prin :

a) — utilizarea unor programe scrise în limbaj de asamblare ;

b) — dacă lungimea realizării de entitate este mai mică decît 32 ko se poate utiliza citirea şi scrierea cu format entitate ;

c) — ştergerea fizică a realizării de entitate care conţine acest inel. Această ştergere trebuie precedată de o operaţie de salvare a valorilor stocate în realizare (se salvează — toate eventualele caracteristici de tip ⟨*inel*⟩ prezente).

Pentru salvarea realizărilor caracteristicii de tip ⟨*referire cu inel*⟩ şi refacerea legăturilor prezentăm în anexa 3 un model de procedură generalizată, adaptabilă la orice tip de relaţie „1 → n". Această procedură utilizează un buffer formatat conform următoarei structuri de formal logic generalizat :

```
FORMAL REFAC-LEGATURI
 DEBUT
   CODANN-ALF   MOT 30 /* codul realizării care conţine inel    */
   CODREF-NUM   MOT 30 /* codul realizării care conţine referire */
```

```
    CODANN-NUM DILATE (1  9)
    CODREF-NUM DILATE (31 9)
FIN
```

Această caracteristică este adaptată salvării celor mai lungi coduri şi/sau numere de realizări din baza de date.

Caracteristicile definite vor fi utilizate cu semnificaţia :

1) cheia realizării entităţii care conţine caracteristica de tip inel este :
- ALF : alfanumerică (CODANN-ALF) ;
- NUM : numerică (CODANN-NUM) ;
- VID : număr de realizare (CODANN-NUM) ;

2) cheia realizării entităţii care conţine caracteristica de tip <referire cu inel> este :
- ALF : alfanumerică (CODREF-ALF) ;
- NUM : numerică (CODREF-NUM) ;
- VID : număr de realizare (CODREF-NUM).

Prin cheie am desemnat caracteristica declarată cheie primară a entităţii. Această procedură permite depanarea tuturor tipurilor de erori detectate.

În afara utilizării procedurii prezentate anterior în funcţie de natura erorii şi de volumul prelucrărilor de efectuat administratorul poate realiza o depanare punctuală a rupturilor de lanţ şi/sau a ciclărilor în lanţ construind programe care înglobează următoarele operaţii :

1 — marcarea adresei realizării entităţii care conţine caracteristica de tip inel implicată într-o variabilă de lucru X_1 ;

2 — parcurgerea secvenţială a tuturor realizărilor entităţii care conţine declaraţia de <referire cu inel> asociată pentru care se efectuează, condiţional, operaţiile :

— pentru fiecare caracteristică care referă acest inel se pune referirea la valoarea nedefinit ;

— se încarcă această referire cu valoarea lui X_1 ;

3 — testarea coerenţei logice a noului ansamblu realizat.

Deşi ansamblul acestor operaţii pare suficient şi pentru depanarea erorilor de tip 3) şi 4) totuşi nu recomandăm utilizarea acestora.

Caracteristici declarate chei de acces

Incoerenţele care pot fi detectate la acest gen de caracteristici pot fi :
- cicluri în lanţurile de sinonime (similar cicluri la inel) ;
- existenţa unor valori ale caracteristicii care nu sînt clasate în dicţionar ;
- rupturi de lanţuri pentru sinonime.

Pentru determinarea coerenţei în cazul caracteristicilor declarate cheie de acces (cu cheie unică sau dublă, ordonată sau neordonată) se execută secvenţa :

```
M Yᵢ = 0  M Yⱼ = 0
FAIRE
POUR TOUT ⟨citaţie entitate₁⟩ PAR ⟨cheie⟩ ;
   M Yᵢ = Yᵢ + 1
   SI Yᵢ > Nmre₁
     ALORS
       I (1) '**CICLARE** LA CHEIA' I (+1) '⟨cheie⟩' ECRIRE
       SORTIE
     FIN
   FIN
FIN
```

```
POUR TOUT <citaţie entitate_1>
   SI EXISTE <cheie>
      ALORS
         M Y_j = Y_j + 1
   FIN
FIN
SI  Y_i > Y_j
   ALORS
   I(1) '** CICLARE ** ÎN LANŢUL' I(+1) 'SINONIMELOR' ECRIRE
   SINON
   SI Y_j > Y_i
      ALORS
   I (1) '** RUPTURI ** IN LANTUL' I (+1) 'SINONIMELOR' SAU'
      I(+1) 'VALORI NECLASATE IN DICTIONAR' ECRIRE
   FIN
  FIN
?
```

Pentru a reface incoerenţele detectate pe ansamblul valorilor asociate unei chei se realizează următoarele operaţii :

1) se parcurg secvenţial realizările entităţii care conţin cheia :
   — se salvează punctual fiecare valoare atribuită cheii (în zone de manevră din entitate, în caracteristici ale unei entităţi de lucru definită pentru astfel de operaţii sau pe un fişier extern bazei de date) ;
   — se şterge intrarea asociată din dicţionar (prin atribuirea valorii (1) ;
2) se parcurg secvenţial realizările entităţii care conţin caracteristica cheie şi valorile salvate se atribuie acestei caracteristici ;
3) se efectuează testele de coerenţă.

Şiruri de biţi

Incoerenţele detectate în cazul şirurilor de biţi se referă la neconcordanţe existente între valoarea contorului şi numărul de biţi poziţionaţi la valoarea 1 în şir.

Pentru fiecare entitate sau caracteristică de tip <invers> se va controla coerenţa şirului de biţi de prezenţă şi a contorului asociat cu secvenţa :

```
M Y_i = D TOUT {<nume invers>  } [<calificare>]
               {<nume entitate>}
M Y_j = 0 M Y_k = 0
FAIRE
   M Y_j = Y_j + 1
   SI Y_j > Nmre
      ALORS
         SORTIE
   FIN
   SI EXISTE UN {<nume invers>  } Y_j [<calificare>]
                {<nume entitate>}
      ALORS:
         M Y_k = Y_k + 1
   FIN
   REFAIRE
FIN
SI Y_i > Y_k
   ALORS
      M Z1 = 'MAI MARE'
   SINON
      SI Y_i < Y_k
         ALORS
```

```
      M Z1 = 'MAI MIC'
   FIN
FIN
SI Y1 ] = Yk
   ALORS
   I(1) '**INCOERENTA** CONTOR' I (+1) Z1
   I(+1) 'DECIT NUMARUL REALIZATORILOR' I(+1) 'PREZENTE LA'
   I (+1) ' {nume entitate} ' ECRIRE
            {nume invers  }
FIN
```

Pentru caracteristicile de tip ⟨*entitate*⟩ depanarea eventualelor erori poate fi efectuată numai cu ajutorul unor secvenţe de program scrise în limbaj de asamblare. Aceste secvenţe pot fi generalizate în sensul furnizării, în variabilele de lucru, a parametrilor de execuţie (adresa virtuală a contorului, valoarea contorului, numărul bitului care trebuie poziţionat/şters etc.). Execuţia lor va fi lansată prin ordine, EXEC... în programe scrise în LMD.

Pentru caracteristicile de tip ⟨*invers*⟩ depanarea eventualelor erori poate fi efectuată sau de maniera celei prezentate pentru ⟨*entitate*⟩ sau prin efectuarea următoarelor operaţii :

1) ştergerea caracteristicii de tip ⟨invers⟩ prin secvenţa :

```
M Y1 = 0
FAIRE
   M Y1 = Y1 + 1
   SI Y1 > Nmre ALORS SORTIE FIN
   SE UN ⟨nume invers⟩ Y1 [⟨calificare⟩]
   REFAIRE
FIN
```

2) generarea caracteristicii de tip ⟨*invers*⟩ prin parcurgerea secvenţială a caracteristicii inversate şi aplicarea filtrului de selecţie, pe valorile realizărilor, care constituie criteriul de inversare.

Suprimarea realizărilor de ⟨entitate⟩

Deoarece gestiunea caracteristicilor de tip ⟨invers⟩ şi a caracteristicilor de tip ⟨*referire simplă*⟩ este în sarcina utilizatorului, acesta este obligat să asigure, înaintea ştergerii realizării de rang Y_1 a unei entităţi, execuţia următoarelor operaţii :

1) dacă entitatea este grupată formal în ⟨*n*⟩ caracteristici de tip ⟨invers⟩ atunci pentru fiecare se va executa secvenţa :

```
SI EXISTE UN ⟨nume invers⟩ Y1 [⟨calificare⟩]
   ALORS
      SE UN ⟨nume invers⟩ Y1 [⟨calificare⟩]
FIN
```

2) dacă realizarea este referită de mai multe realizări a ⟨r⟩ entităţi distincte atunci pentru fiecare entitate se va executa secvenţa :

```
M X1 = UN   ⟨nume entitate⟩ Y1 [⟨calificare⟩]
POUR TOUT ⟨nume entitate ⟩ [⟨calificare⟩]
   SI ⟨referire simplă⟩ [⟨calificare⟩] = X1
      ALORS
         M ⟨referire simplă⟩ [⟨calificare⟩] = U
   FIN
FIN
```

3) ştergerea efectivă a realizării Y_i cu instrucţiunea S UN ⟨*nume entitate*⟩ Y_i [⟨*calificare*⟩] sau S X_i, dacă X_i desemnează formal şi real realizarea implicată.

Reprezentarea relaţiilor de tip „$m \leftrightarrow n$"

Acest tip de relaţii nu sînt acceptate în această formă de SGBD-SOCRATE (în particular nici de alte SGBD-uri) de aceea vor fi transformate în relaţii elementare de tip „$1 \rightarrow n$".

Dacă avem două entităţi X şi Y aflate în relaţia „$m \leftrightarrow n$" (o realizare a entităţii X este posesor pentru mai multe realizări din Y şi o realizare a entităţii Y este posesor pentru mai multe realizări din X), această relaţie va fi transformată în relaţii elementare, introducînd o entitate de legătură L, aflată în relaţia „$1 \rightarrow n$" cu entitatea X şi entitatea Y, prin intermediul căreia se realizează legăturile.

Schematic, transformarea relaţiei „$m \leftrightarrow n$" în relaţii „$1 \rightarrow n$" se prezintă ca în figura 8.18.

Această schemă este generalizată în sensul că X şi Y pot desemna aceeaşi entitate (realizările entităţii sînt în relaţia „$m \leftrightarrow n$").

Structura conceptuală a entităţii de legătură L este :

```
ENTITE ⟨Nmre₁⟩ ⟨identificator enti. leg.⟩
  DEBUT
    ⟨referire Xy⟩
    ⟨referire Yx⟩
    [⟨cod entitate X⟩]
    [⟨cod entitate Y⟩]
    [⟨alte informaţii⟩]
  FIN
```

⟨$Nmre_1$⟩ : numărul maxim de realizări al entităţii de legătură. Acest număr se stabileşte cu formula :

⟨$Nmre_1$⟩ : = (⟨Nmre x⟩ * ⟨Nmre y⟩) * k

unde k este un coeficient de reducere a cărui valoare va fi stabilită de proiectant în funcţie de proprietăţile datelor ;

⟨*referire* X_y⟩ : identificatorul caracteristicii de tip referire cu inel care grupează pentru entitatea X realizări ale entităţii y.

⟨*cod entitate* x⟩ : identificatorul unei caracteristici destinată memorării valorii cheii primare a realizării entităţii x (sau a numărului de realizare) aflată în relaţie.

Natura şi structura acestei caracteristici este :

– de tipul caracteristicii corespondente din entitatea x, dacă se utilizează cheia ca element de identificare ;

– de tip valoare numerică cu plaja de variaţie a valorii în intervalul [1, ⟨Nmre x⟩], dacă se utilizează numărul de realizare al entităţii, ca element de identificare.

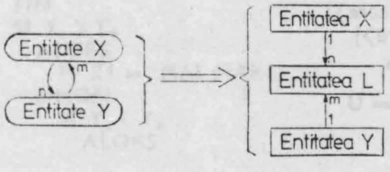


Fig. 8.18. Transformarea relaţiei „m ↔ n" în relaţii elementare „1 → n"

Deşi aceste elemente sînt redundante, cu caracteristicile de tip referire corespondente, prin introducerea lor se asigură un control logic al coerenţei grupajelor. La momentul creerii realizărilor acestei entităţi se va asigura încărcarea acestor caracteristici în momentul construirii relaţiei (încărcarea referirilor).

Deşi această entitate a fost introdusă pentru a permite în principal realizarea fizică a relaţiilor de tip „*m* ↔ *n*" nu este exclusă şi definirea altor tipuri de caracteristici (alte informaţii) a căror gestiune este în sarcina programelor construite de utilizator.

De exemplu, dacă dorim să reprezentăm relaţia „produs-materii prime" rezumată prin: „la fabricarea unui anumit produs sînt necesare mai multe materii prime" şi „o materie primă poate fi utilizată la fabricarea mai multor tipuri de produse", care este de tip „*m* ↔ *n*", entitatea de legătură asociată poate conţine informaţii referitoare la consumul specific de materie primă, tehnologia de prelucrare etc.

Pentru a ilustra modul de realizare a relaţiei „*m* ↔ *n*" prin entitatea de legătură (fig. 8.19) considerăm faptul că realizarea {X1} se află în relaţie cu realizările {Y1, Y3, Y5} iar realizarea {Y1} se află în relaţie cu realizările {X1, X2, X3}.

În această schemă s-au utilizat notaţiile:

X_i : realizarea *i* a entităţii X ;

INFO (X_i) — informaţiile realizării (diferite de legături) ;

∂X_i — adresa virtuală a realizării X_i ;

00 — zero binar (terminator stivă).

În conformitate cu această schemă, pentru a afla care sînt realizările entităţii Y aflate în relaţie cu realizarea X1 se parcurge inelul definit în realizarea X1, iar prin referirile la realizările entităţii Y (relaţia „1 – 1" cu capul de lanţ) devin accesibile informaţiile INFO (Y_i) ale acestora.

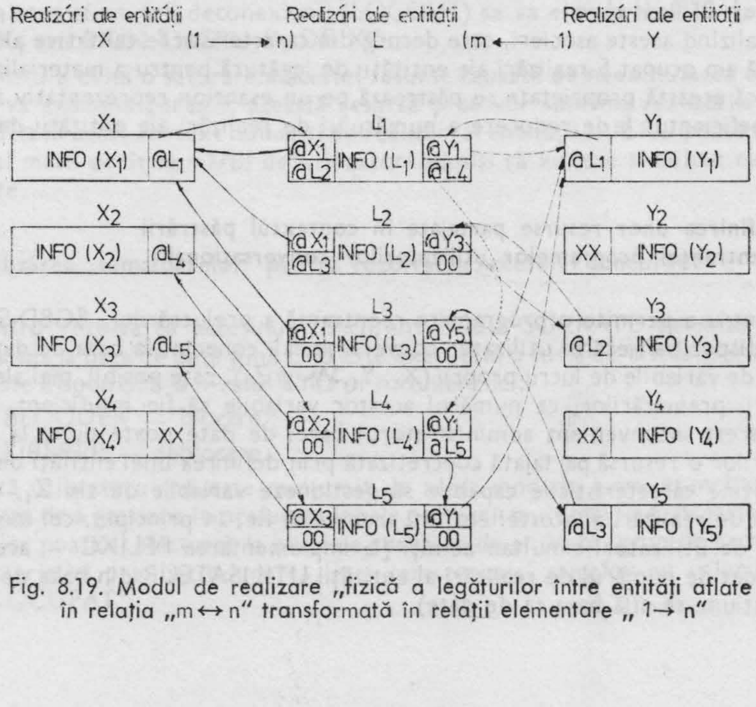


Fig. 8.19. Modul de realizare „fizică a legăturilor între entităţi aflate în relaţia „m ↔ n" transformată în relaţii elementare „ 1→ n"

Parcurgerea poate fi ilustrată astfel:

$\partial L_1 : \partial Y_1$ INFO(Y1)

$\partial L_2 : \partial Y_3$ INFO(Y_3)

$\partial L_3 : \partial Y_5$ INFO(Y_5)

⟨*inel*⟩ : ⟨*referire*⟩ → ⟨*informaţii*⟩

Precizăm faptul că la o relaţie de tip „*m* ↔ *n*", transformată, între entităţile X şi Y furnizarea datelor necesare grupării realizărilor din Y la o anumită realizare din X permite şi construirea relaţiei existente între fiecare dintre realizările entităţii Y şi realizarea din X. Programatorul de aplicaţie trebuie să prevadă, înaintea execuţiei secvenţelor de generare a unei legături, teste de validare a absenţei acesteia (testele se pot aplica sau asupra caracteristicilor de tip referire sau asupra caracteristicilor destinate reţinerii cheilor primare ale realizărilor şi/sau a numerelor de realizare din entitatea de legătură).

Dacă notăm cu $C(X_i)$ valoarea cheii realizării *i* a entităţii X, atunci dacă secvenţa de asociere (A) este:

1) pentru X_y: $C(X_1)$ A $C(Y_1)$, $C(Y_3)$, $C(Y_5)$, atunci vor fi realizate legăturile:
   – X_1 grupează realizările $\{Y_1, Y_3, Y_5\}$ prin $\{L_1, L_2, L_3\}$;
   – pentru Y_1, Y_3 şi Y_5 se vor realiza legăturile cu X_1 prin L_1, L_2 şi respectiv L_3;
2) pentru Y_x : $C(Y_1)$ A $C(X_1)$, $C(X_2)$, $C(X_3)$, atunci se grupează X_2 şi X_3 prin L_4 şi L_5 (X_1 este grupat deja din secvenţa 1).

După efectuarea asocierilor specificate se vor realiza grupajele:

– X_1 $\{Y_1, Y_3, Y_5\}$ prin $\{L_1, L_2, L_3\}$;
– X_2 $\{Y_1\}$ prin $\{L_4\}$;
– X_3 $\{Y_1\}$ prin $\{L_5\}$;
– Y_1 $\{X_1, X_2, X_3\}$ prin $\{L_1, L_4, L_5\}$;
– Y_3 $\{X_1\}$ prin $\{L_2\}$;
– Y_5 $\{X_1\}$ prin $\{L_3\}$.

Analizînd aceste asocieri, care decurg din caracteristicile calitative ale datelor, rezultă că am ocupat 5 realizări ale entităţii de legătură pentru a materializa 10 asocieri. Dacă această proprietate se păstrează pe un eşantion reprezentativ al datelor atunci coeficientul *k* de reducere a numărului de realizări ale entităţii de legătură va fi 0,5.

Definirea unor resurse partajate în contextul păstrării reentranţei programelor utilizatorilor conversaţionali

Pentru a permite o programare reentrantă a prelucrărilor, SGBD-SOCRATE pune la dispoziţia fiecărui utilizator conversaţional, conectat la baza de date, un set complet de variabile de lucru proprii (X_i, Y_i, W_i şi Z_i). Este posibil, mai ales în cazul înlănţuirii prelucrărilor, ca numărul acestor variabile să fie insuficient. Pentru a elimina acest inconvenient administratorul bazei de date poate pune la dispoziţia utilizatorilor o resursă partajată concretizată prin definirea unei entităţi de LUCRU, care conţine caracteristicile capabile să gestioneze variabile de tip Z_i, W_i şi Y_i. Numărul de realizări al acestei entităţi trebuie să fie, în principiu, cel mult egal cu numărul de utilizatori simultan admişi (la implementarea FELIXC – acest număr poate fi dat de numărul de realizări al entităţii UTILISATEUR din baza de baze în a cărei gestiune se află baza sa de date).

Structura de principiu a acestei entităţi este de forma :

ENTITE ⟨Nmre⟩ LUCRU
DEBUT
WZ1 MOT /* pentru variabile de */
.
.
.
WZ_i MOT /* tip Zi */
WY_1 DE $-(2^{31}-1)$ A $+(2^{31}-1)$ /* pentru variabile de */
WY_i DE $-(2^{31}-1)$ A $+(2^{31}-1)$ /* tip Y_i */
[WW1 DECIMAL ... WW_i ...] /* pentru variabila W_i dacă implementarea admite */
FIN

Protocolul de utilizare a realizărilor acestei entităţi poate fi următorul :

1) — la începutul fiecărei prelucrări sau sesiune utilizatorul îşi rezervă o realizare prin execuţia unei macroinstrucţiuni care conţine instrucţiunile :

BLOQUER
G UN LUCRU X_i DE X0
LIBERER

2) — dacă rezervarea se execută la început de sesiune, pentru întreaga sesiune, atunci dacă sînt apelate mai multe programe succesive care utilizează realizarea rezervată este necesară transmiterea adresei acesteia. Adresa poate fi transmisă prin rezervarea variabilei X_i ca adjectiv de desemnare sau prin conservarea numărului ei de realizare într-o variabilă Y_i cu cererea MY_i=NUMDE X_i. Utilizarea unei forme sau a celeilalte va fi la latitudinea utilizatorului. Variabilele definite vor fi citate prin Wxi DE X_i sau Wzi DE UN LUCRU Y_i DE XO în funcţie de opţiunea utilizatorului ;

3) — înaintea efectuării deconexiunii (LOGOUT) se va elibera această resursă prin S X_i sau S UN LUCRU Y_i DE XO ;

4) — pentru a evita o saturare a acestei resurse cauzată de neexecutarea cerinţei 3) se va examina periodic această resursă şi se vor suprima realizările definite. Privit în acest context numărul de realizări al entităţii LUCRU poate fi cu mult mai mare decît numărul de utilizatori admişi să lucreze simultan cu baza de date.

Utilizarea „semafoarelor" pentru rezolvarea accesului concurent

Pentru rezolvarea accesului concurent SGBD SOCRATE permite blocarea oricărui acces la baza de date pînă cînd este executată o suită de instrucţiuni. Această blocare este disponibilă la nivelul LMD prin comenzile :

— BLOQUER — blocare ;

— LIBERER — deblocare.

Dacă utilizatorul nu este mulţumit de acest mod de lucru atunci acesta are posibilitatea de a gestiona în prelucrările sale protecţii mult mai fine. Această gestiune a resurselor poate fi efectuată la nivel de realizare de entitate, grup de caracteristici sau caracteristici prin introducerea unei caracteristici semafor cu valorile binare „LIBER—OCUPAT".

Toate prelucrările care lucrează cu elementul respectiv trebuie să respecte protocolul :

a) dacă semaforul este în starea „LIBER" :

— se trece în starea „OCUPAT" sub controlul comenzilor BLOQUER...LIBERER ;

— se execută prelucrările dorite (accesul la baza de date al celorlalţi utilizatori nu mai este blocat) ;

— se eliberează resursa prin poziţionarea semaforului, sub controlul comenzilor BLOQUER...LIBERER, în starea „LIBER" ;

b) dacă semaforul este în starea „OCUPAT" :

— se semnalează un mesaj, utilizatorului, care anunţă această stare şi se pune prelucrarea în aşteptare.

Utilizatorul de la terminal are posibilitatea :

— să abandoneze prelucrarea ;

— să o pună în aşteptare şi să o reia după un anumit timp.

Ambele posibilităţi pot fi incluse în logica internă a programului de cereri pentru a fi executate în mod automat.

Definirea unor resurse globale pentru o bază de date

Dacă facem o comparaţie cu limbajele algoritmice care solicită rezervarea oricărei variabile înaintea utilizării sale acest mod de lucru a fost păstrat şi la LMD SOCRATE. Diferenţa între LMD şi acest tip de limbaje este dată de faptul că toate caracteristicile descrise în structură reprezintă „variabile globale", în accepţiunea acestei noţiuni utilizate de limbajele algoritmice. Problema pe care dorim să o tratăm este aceea de a defini astfel de „variabile globale" a căror modificare să nu se repercuteze asupra datelor asociate modelului descris de structură.

Un exemplu de astfel de variabile globale sînt caracteristicile RĂSPUNS, RASP, CONTINUATI etc. descrise în schema conceptuală prezentată în capitolul 3.

Aceste variabile au un identificator cu semantică clară pentru utilizator la momentul solicitării sale într-o prelucrare conversaţională. Solicitarea identificatorului utilizează avantajele afişării standard la cererile de comunicare cu exteriorul (M... = EXT) iar calificarea necesară este concisă (M ⟨*identificator*⟩ DE XO = EXT).

Acestei afirmaţii i se poate aduce un contraargument : de ce să nu folosim o variabilă de lucru căreia să i se asocieze un text la editare ? Putem afirma că soluţia propusă de noi este mai bună din următoarele motive :

1) dacă programele incorporează în cadrul lor texte definite ca literali atunci orice modificare a formei acestor literali necesită :

— modificarea acestui text în toate programele care îl utilizează. Aceste programe pot fi, pentru o bază de date, păstrate sub forma :

• cod sursă în fişiere de tip „fişier de intrare standard" ;

• macroinstrucţiuni şi/sau programe precompilate.

— recatalogarea codului macroinstrucţiunilor şi programelor precompilate afectate de modificare.

Un exemplu de modificare ar fi transformarea formei de prezentare externă RĂSPUNS : într-o formă imperativă RĂSPUNDEŢI :.

2) introducerea unor literali „în program a căror evoluţie este incertă provoacă o legare a programelor de date" (de aceşti literali în acest caz).

În cazul utilizării unor variabile globale acţiunile realizate de utilizator pot fi rezumate în cazul modificării lor, pentru exemplul nostru, astfel :

1) se defineşte o macroinstrucţiune RĂSPUNS

Exemplu :

: DEFMAC RĂSPUNS : EXP RĂSPUNDEŢI : FDEF ?

2) această macroinstrucţiune se va propaga în mod automat în toate programele de tip „fişier de intrare standard" sau macroinstrucţiune, la momentul apelului acestora ;

3) este necesară numai recatalogarea programelor precompilate care o citează.

Mai mult, acest gen de „variabile globale" pot beneficia de restricţiile de validare standard definite la momentul declarării lor. Aceste variabile pot fi definite în orice moment utilizînd facilitatea de „adăugare la structură" a unor noi atribute.

Ca exemplu de validare, variabila CONTINUAŢI acceptă răspunsurile DA, NU, Y (yes) şi N (not), orice alt răspuns (diferit de U, /, NON, ⟨cr⟩ care sînt răspunsuri ale dialogului conversaţional standard) va fi refuzat solicitînd introducerea valorii.

Condiţiile de validare pot fi modificate prin redefinirea caracteristicii. De exemplu dacă dorim ca variabila să accepte numai răspunsurile Y şi N vom redefini structura modificînd caracteristica astfel : CONTINUAŢI (4 2) (Y N), ceea ce nu va avea nici-o implicaţie asupra programelor care o citează indiferent de formatul lor de prezentare.

În cadrul utilizării unor „literali" la afişare răspunsul trebuie validat de programator prin logica inclusă în programul său, logică care poate să „cadă" la modificări.

Pentru definirea acestui tip de variabile indicăm ca ele să fie grupate pe **pagini** iar fiecare grup să fie aliniat la frontieră de **pagină**. Cu acest mod de definire se asigură prezenţa lor, la citare, într-un cadru de lucru destinat numai acestor variabile. Avantajul obţinut este acela că prin frecvenţa sa de utilizare cadrul respectiv va fi practic rezident în memoria centrală. Din acest considerent definirea acestor variabile a fost făcută la începutul blocului asociat spaţiului virtual „FICHIER", în cadrul exemplului nostru.

Generarea automată a programelor de întreţinere şi/sau exploatare a bazei de date

Consideraţii preliminare

Introducerea automatizării programării [AF1] apare din aceeaşi necesitate care a dus anterior la utilizarea şi introducerea tehnicii de calcul. Aceasta reprezintă unul din paşii care trebuiesc parcurşi în vederea unei reale industrializări a programării. Programarea automată poate fi definită ca totalitatea tehnicilor şi metodelor utilizate pentru obţinerea automată, prin generarea cu ajutorul calculatorului, a programelor şi procedurilor de prelucrare, necesare unei aplicaţii funcţionale. Programele rezultate din generarea automată sînt prezentate în format sursă şi au ca suport un limbaj gazdă, în general un limbaj universal, de nivel înalt. Nu este exclusă nici posibilitatea

generării programelor pentru un limbaj specific de tip asamblor, limbaj de comandă al unui sistem de operare etc. Produsele generate sînt portabile în măsura în care limbajul gazdă suport este portabil.

Programele generate automat se aduc la un format executabil efectuînd asupra lor operaţiile cerute de sistemul de operare gazdă (de exemplu : compilare, asamblare ediţie de legături etc.) sau de software-ul suport pe care se grefează. Codul generat în format sursă poate fi modificat de către programator constituind astfel un suport pentru dezvoltarea programelor de aplicaţie.

Metode de generare automată

Metodele investigate, în domeniul generării automate, pot fi materializate prin :

1) Construirea unui produs software care primind la intrare informaţii despre :
- structura (şi descrierea) datelor de intrare,
- structura (şi descrierea) datelor de ieşire,
- tipurile şi algoritmii operaţiilor executate asupra datelor pentru obţinerea ieşirilor,

să construiască programele specifice acestor prelucrări.

Acest produs software poate fi construit ca o prelungire firească a unui alt produs de proiectare asistată cu calculatorul (de exemplu pornind de la datele furnizate în urma utilizării produsului PACSIN) sau independent.

În cazul utilizării rezultatelor PACSIN produsul devine, în general, cu un grad de portabilitate scăzut, chiar dacă este realizat într-un limbaj de nivel înalt (va avea cel mult gradul de portabilitate al acestuia).

Pentru utilizarea rezultatelor PACSIN (structura conceptuală descrisă cu LDD, algoritmii de prelucrare, structura logică a intrărilor etc.) o metodă ar consta în :

— construirea unor macroinstrucţiuni sintactice cu ajutorul LPS ;

— numele acestor macroinstrucţiuni vor fi reprezentate de cuvintele cheie SOCRATE ;

— formarea, din forma sursă a structurii, a unui program sursă LPS cu apeluri la aceste macroinstrucţiuni. Parametrii de apel vor fi reprezentaţi de descriptorii asociaţi caracteristicilor (identificator, limită inferioară, limită superioară, declaraţie de acces etc.) ;

— compilarea acestui program şi obţinerea sursei programelor de exploatare. Aceste programe sînt generate conform combinărilor efectuate în macroinstrucţiunile sintactice.

Generarea se poate desfăşura parametric în sensul că se pot realiza coduri sursă pe game de operaţii.

Produsul ar putea construi, conform acestei idei, un mediu atomic de programare pentru baza de date (citaţii de elemente individuale a căror combinare ar permite obţinerea unui program specific).

Construirea unui produs software, independent de produsele existente, sută la sută portabil.

În [A1] acest produs, proiectat logic şi realizat fizic parţial denumit SS & PADT (SOCRATE STRUCTURE & Programming Design Tool) ar trebui să dispună de următoarele componente :

— SDT (Structure Design Tool) ;

– SDO (Structure Definition Optimizer) ;
– PDT (Program Design Tool) ;
– POC (Program Optimizer Cod) ;
– RSDT (Relational Structure Design Tool) ;
– RPDT (Relational Programming Design Tool).

Produsul este sută la sută portabil SOCRATE V 1.6. În afara componentelor SDO și POC a căror acțiune este lansată conversațional dar se desfășoară tip „batch processing" restul componentelor sînt „help-by-help", nivelul de „help" fiind un parametru modificabil dinamic.

SDT permite descrierea asistată a structurilor gestionate cu SGBD SOCRATE. Utilizatorul poate interveni în orice moment pentru a vizualiza structura definită, a efectua modificări în ea, a vizualiza starea de ocupare a spațiului virtual sau a solicita asistența calculatorului. Dacă structura definită este validată de utilizator atunci se construiește în mod automat un tezaur de citații elementare. Acest tezaur va fi utilizat pe parcursul PDT pentru asistarea acestei operații.

PDT permite construirea asistată a programelor de exploatare. Elementele manipulate de program pot fi introduse direct de utilizator, caz în care se verifică corectitudinea sintactică și semantică a citațiilor, sau vor fi selectate prin defilarea pe ecran a caracteristicilor aparținînd unui (unor) domeniu (domenii) ales(e) de utilizator.

Dacă utilizatorul se fixează asupra unui obiect atunci i se solicită ce tip de operație(i) dorește să execute asupra acestuia. În funcție de tipul obiectului și al operației se generează un cod în LMD specific.

Rezultatul generării unui program este catalogat, sub un nume definit de utilizator, nume care trebuie să fie unic.

Dacă există un program cu numele solicitat de utilizator PDT semnalează acest lucru utilizatorului acesta putînd realiza, de exemplu, modificări tip „editor de texte" asupra codului sursă asociat.

În cazul introducerii directe a elementelor de către utilizator se verifică corecsitudinea identificatorilor și a citațiilor asociate acestora. Dacă citația este incorectă tau utilizatorul nu o introduce atunci i se prezintă citațiile alternative ale elementului solicitînd alegerea uneia dintre acestea.

SDO realizează o optimizare a structurii obținute conform specificațiilor versiunii de SGBD pentru care se generează și conform criteriilor de optimizare prezentate anterior în acest capitol.

POC realizează optimizarea codului sursă al programului rezultat aplicînd o tehnică de rafinare a acestuia.

RSDT și RPDT, mai puțin dezvoltate actual, vor permite o interfață relațională cu utilizatorul. Acesta va opera cu structura relațională a bazei de date cu ajutorul unui limbaj relațional. Rezultatul generării, din descrierea relațională și programele de exploatare va fi reprezentat de o structură conceptuală SOCRATE și respectiv programe de cereri exprimate în LMD.

Pentru toate componentele rezultatul acțiunii lor poate fi comunicat unei implementări a SGBD SOCRATE sub forma unui fișier de intrare standard. Acest fișier va conține atît comenzile exprimate în limbajul de comandă al sistemului de operare cît și textul exprimat în LDD sau LMD SOCRATE. Singura acțiune de executat, din partea utilizatorului, asupra acestor fișiere va fi reprezentată de lansarea lor în execuție pentru baza de date pe care o exploatează.

2) Construirea unor generatoare de programe specializate pe clase de operații ca : creere, actualizare, ștergere, listare, analiză și interogare etc.

Această metodă este ușor aplicabilă în momentul în care se ajunge la o formalizare a descrierii intrărilor și ieșirilor de date. Programele pot fi generate în bloc (pentru toate tipurile de operații) pornind de la un set limitat de parametri. Setul de parametri combinat cu formalizarea descrierii, formalizare inclusă în general în logica internă a generatorului, va duce la obținerea unor programe variate din punct de vedere funcțional și adaptate perfect unei probleme specifice.

Vom dezvolta mai mult această metodă deoarece, deocamdată, este disponibilă oricărui utilizator.

3) Utilizarea combinată a celor două metode prezentate anterior.

Construirea generatoarelor de programe specializate pe clase de programe

Formalizarea descrierii structurii conceptuale

Structura conceptuală a unei baze de date (BDS), din domeniul economic, realizată cu SGBD SOCRATE se prezintă, văzută prin prisma LDD, ca un ansamblu de relații între entități și/sau elemente aparținînd acestor entități.

Această afirmație poate fi rezumată, utilizînd notația BNF, astfel :

⟨*structură conceptuală* BDS⟩ : : = ⟨*relații de agregare*⟩,
(⟨*entități nomenclator*⟩,
⟨*entități utilizare*⟩,
⟨*entități documentare*⟩,
⟨*entități nucleu*⟩)

Entitățile care compun structura conceptuală, privite prin prisma conținutului informațional și a modului de utilizare în programele de exploatare, pot fi grupate în următoarele clase :

— entități de tip nomenclator (dicționar) ;
— entități de tip documentare ;
— entități de tip utilitare ;
— entități de tip nucleu.

Pentru exemplificarea modului de utilizare practică a noțiunilor introduse în continuare se va consulta schema conceptuală descrisă în capitolul 3.

Entitățile de tip nomenclator sînt „externe" celorlalte tipuri de entități și conțin datele în clar aferente informațiilor codificate în acestea și eventual grupări ale realizărilor acestor entități. Între realizările acestor tipuri de entități pot exista diverse relații de grupare care, privite împreună cu relațiile stabilite cu realizările entităților de tip nucleu, definesc reale subdomenii de date și includ diverse căi de parcurgere a acestora.

Entitățile de tip documentare conțin datele cu un anumit grad de repetabilitate temporală aferente unei realizări de tip nucleu (istoric al evoluției unor date). Datele conținute în acest tip de entitate pot da naștere la relații de agregare cu celelalte tipuri de entități.

Entitățile de tip utilitare au rolul de a facilita activitățile de programare și/sau regăsire a datelor și includ date pentru : materializarea relațiilor de tip „$m \leftrightarrow n$", utilizarea resurselor partajate, definirea unor mecanisme de acces, etc.

Entitățile de tip nucleu conțin datele de bază ale domeniului (subdomeniului) care constituie obiectul principal al sistemului (subsistemului) respectiv.

Relaţiile de agregare, stabilite între entităţile care definesc modelul structural, pun în evidenţă caracteristicile calitative ale datelor, de interdependenţă-intercondiţionare, şi pot fi grupate în următoarele tipuri : implicite, explicite şi transparente.

Acest grupaj poate fi ilustrat în notaţia BNF astfel :

⟨relaţii de agregare⟩ : : = *⟨implicite⟩*/*⟨explicite⟩*/*⟨transparente⟩*

Relaţiile de agregare implicite decurg din modul de descriere a modelului structural şi sînt reprezentate de entităţi imbricate, blocuri şi caracteristici elementare descrise în cadrul entităţilor. Modul de producere a acestui tip de relaţii este modul natural de descriere a structurilor arborescente.

Relaţiile de agregare explicite decurg din descrieri explicite şi pot fi interpretate ca relaţii de agregare „forţate". Aceste relaţii permit punerea în evidenţă a reţelelor de date şi gruparea datelor pe diverse caracteristici structurale (legături de tip inel-referire).

Relaţiile de agregare transparente decurg din descrierea naturală a unor caracteristici identice, din punct de vedere semantic, în entităţi diferite.

Dintre tipurile de entităţi prezentate se poate stabili un standard de formalizare pentru entităţile de tip nomenclator şi pentru entităţile utilitare utilizate pentru materializarea relaţiilor de tip „$m \leftrightarrow n$".

Formatul general al unei entităţi de tip nomenclator este :

ENTITE ⟨Nmre⟩ *⟨identificator enti.⟩*
[*⟨cod⟩*]
[*⟨identificatori inel⟩*]
[*⟨denumire⟩*]
[*⟨referinţe⟩*]
[*⟨alte informaţii⟩*]
FIN

unde :

— ⟨Nmre⟩ : numărul maxim de realizări al entităţii ;
— *⟨identificator enti.⟩* : numele atribut entităţii de către utilizator ;
— *⟨cod⟩* : identificatorul şi tipul caracteristicii de tip cod (cheie primară utilizată la identificarea realizărilor entităţii). Poate fi de tip numeric (NUM), alfanumeric (ALF) sau poate lipsi (VID) în cazul în care există identitate între conţinutul său şi numărul de realizare al entităţii ;
— *⟨identificatori inel⟩* : identificatorii caracteristicilor de tip *⟨inel⟩* care definesc relaţii între entităţile de tip nomenclator ;
— *⟨denumire⟩* : identificatorii zonelor care conţin denumirea în clar a realizării cu codul *⟨cod⟩* ;
— *⟨referinţe⟩* : identificatorii caracteristicilor de tip *⟨referire⟩* definite între entităţile de tip „nomenclator". Aceste referiri permit evidenţierea legăturilor existente între entitatea în cauză şi alte entităţi nomenclator sau între o realizare a entităţii şi anumite realizări ale aceleiaşi entităţi. Caracteristicile de tip *⟨referire⟩* pot fi simple sau cu inel asociat ;
— *⟨alte informaţii⟩* : caracteristicile descrise în cadrul acestora nu ridică probleme la formalizare.

Precizăm faptul că ordinea de definire a caracteristicilor entităţii este aleasă de proiectant în funcţie de raţiuni care privesc optimizarea încărcării realizărilor şi a accesului la realizări. Formalizarea nu introduce restricţii asupra modului de aranjare a caracteristicilor şi a tipului asociat acestora. Pentru a obţine o automatizare a pro-

gramării, cu un efort minim din partea utilizatorului, este necesar, în această formalizare, prezenţa anumitor identificatori construiţi în conformitate cu reguli de producţie precise. Dacă construirea acestor identificatori se realizează liber atunci numărul de parametrii furnizaţi produsului de generare creşte direct proporţional cu gradul de libertate ales. Deoarece nu există restricţii asupra modului de descriere a caracteristicilor entităţii definite prin noţiunea de ⟨*alte informaţii*⟩ acestea nu vor fi utilizate în regulile de descriere a tipurilor de modele de descriere deduse din modelul general de formalizare.

Pentru acest model general de formalizare sînt prezentate regulile de producere, sub formă de automate, în figura 8.20.

Pentru exemplificarea modului de deducere a unui tip de model considerăm că entitatea noastră poate să aibă maxim două caracteristici de tip ⟨*referire*⟩, denumite ⟨ref_1⟩ şi ⟨ref_2⟩, şi cîmpul ⟨*denumire*⟩ conţine maxim două atribute, ⟨den_1⟩ şi ⟨den_2⟩.

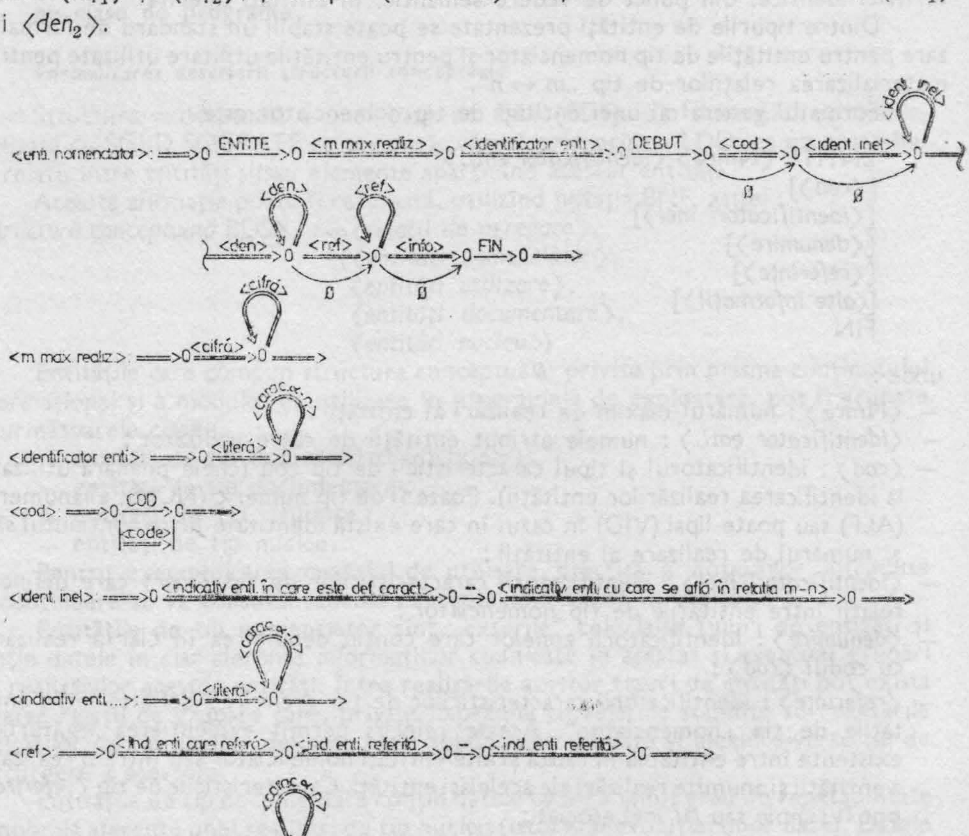


Fig. 8.20. Automatele descrierii entităţilor de tip nomenclator

Cu aceste precizări, descrierea structurală a unei entităţi de tip nomenclator este :

⟨entitate nomenclator⟩ : : =

[⟨*cod*⟩] [⟨*identificatori inel*⟩] ⟨*den*$_1$⟩/ (1)
[⟨*cod*⟩] [⟨*identificatori inel*⟩] ⟨*ref*$_1$⟩ ⟨*den*$_1$⟩/ (2)
[⟨*cod*⟩] [⟨*identificatori inel*⟩] ⟨*ref*$_1$⟩ ⟨*ref*$_2$⟩ ⟨*den*$_1$⟩/ (3)
[⟨*cod*⟩] [⟨*identificatori inel*⟩] ⟨*den*$_1$⟩ ⟨*den*$_2$⟩/ (4)
[⟨*cod*⟩] [⟨*identificatori inel*⟩] ⟨*ref*$_1$⟩ ⟨*den*$_1$⟩ ⟨*den*$_2$⟩/ (5)
[⟨*cod*⟩] [⟨*identificatori inel*⟩] ⟨*ref*$_1$⟩ ⟨*ref*$_2$⟩ ⟨*den*$_1$⟩ ⟨*den*$_2$⟩ (6)

Pentru cazul în care realizările entităţii vor fi prelucrate în „batch processing" sau în conversaţional utilizînd ecrane formatate (video-formate) fiecărui tip de model i se asociază un format standard de introducere a datelor. Acest format este definit în mod unic pentru toate operaţiile de întreţinere a realizărilor (creare, adăugare, ştergere, modificare). Selecţia uneia din aceste operaţii va fi efectuată în funcţie de conţinutul cîmpurilor din linia de intrare şi conţinutul atributelor corespondente, din entitate, la momentul prelucrării. Precizăm, şi în acest caz, că ordinea de definire a cîmpurilor din format, a structurii şi naturii acestora este în sarcina proiectantului. Produsul de generare va recunoaşte o anume structură în funcţie de tipul de model al entităţii specificat de utilizator.

Dacă utilizatorul doreşte să genereze numai programe conversaţionale pentru realizarea operaţiilor de creere, ştergere, actualizare şi adăugare a realizărilor entităţii atunci descrierea caracteristicii(lor) de tip formal care se asociază unui model nu mai este necesară.

Cu aceste precizări descrierea de tip ⟨*formal*⟩ asociată unui model de entitate este :

⟨*macheta nomenclator*⟩ : : =

⟨*ident*⟩ ⟨*cod*⟩ ⟨*den*$_1$⟩/ (1)
⟨*ident*⟩ ⟨*cod*⟩ ⟨*ref*$_1$⟩ ⟨*den*$_1$⟩/ (2)
⟨*ident*⟩ ⟨*cod*⟩ ⟨*ref*$_1$⟩ ⟨*ref*$_2$⟩ ⟨*den*$_1$⟩/ (3)
⟨*ident*⟩ ⟨*cod*⟩ ⟨*den*$_1$⟩ ⟨*den*$_2$⟩/ (4)
⟨*ident*⟩ ⟨*cod*⟩ ⟨*ref*$_1$⟩ ⟨*den*$_1$⟩ ⟨*den*$_2$⟩/ (5)
⟨*ident*⟩ ⟨*cod*⟩ ⟨*ref*$_1$⟩ ⟨*ref*$_2$⟩ ⟨*den*$_1$⟩ ⟨*den*$_2$⟩ (6)

unde :

- ⟨*ident*⟩ : identificatorul formatului de prelucrare ;
- ⟨*cod*⟩ : codul/numărul realizării entităţii ;
- ⟨*ref*$_1$⟩, ⟨*ref*$_2$⟩ : codul/numărul realizării entităţii la care se face referire ;
- ⟨*den*$_1$⟩, ⟨*den*$_2$⟩ : denumirea asociată lui ⟨*cod*⟩.

Precizăm faptul că în aceste formate pot fi descrise şi cîmpurile destinate atributelor entităţii prezentate sub numele ⟨*alte informaţii*⟩.

În figura 8.21 sînt prezentate automatele regulilor de producere pentru conceptul ⟨*macheta nomenclator*⟩.

Pentru entităţile de legătură formatul de structură conceptuală generală a fost prezentat în paragraful anterior.

În cazul în care realizările acestei entităţi vor fi prelucrate în conversaţional utilizînd ecrane formatate sau în „batch processing" acestei structuri i se asociază un format standard (⟨*macheta enti* X_y⟩), astfel :

⟨*macheta enti.* X_y⟩ : : = ⟨*cod macheta*⟩ ⟨*ident. cod enti.* X⟩ ⟨*tip operaţie*⟩(⟨*ident cod enti.* Y⟩ [⟨*info. enti. leg.*⟩])

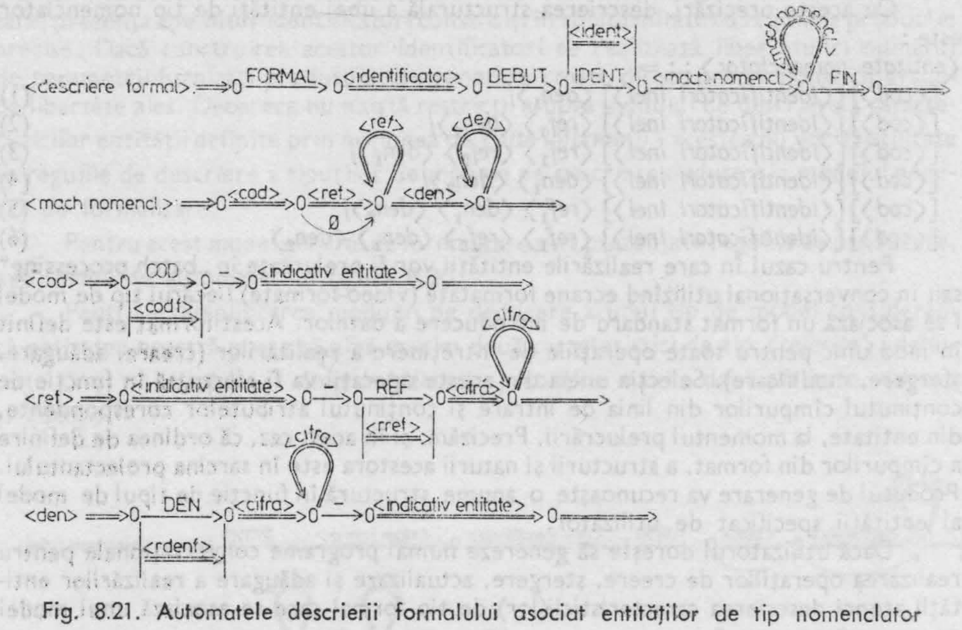


Fig. 8.21. Automatele descrierii formalului asociat entităţilor de tip nomenclator

unde :

— prin (...) am precizat că elementele incluse se pot repeta de un număr nedeterminat de ori. Acest număr este limitat de dimensiunea elementelor repetabile şi dimensiunea liniei de introducere a datelor ;

— ⟨*macheta enti.* X_y⟩ : identificatorul formatului de preluare a datelor necesare grupării realizărilor entităţii Y la o realizare a entităţii X ;

— ⟨*cod machetă*⟩ : identificatorul zonei, din linia de intrare, destinate recepţionării codului machetei ;

— ⟨*ident. cod enti.* X⟩ : identificatorul zonei din linia de intrare care conţine codul/numărul realizării entităţii X la care se vor grupa realizările entităţii Y ;

— ⟨*cod enti.* Y⟩ : identificatorul zonei (zonelor) din linia de intrare care va conţine codul realizării din entitatea Y care se grupează la realizarea desemnată din entitatea X ;

— ⟨*info. enti. leg.*⟩ : structura acestei zone este stabilită de ABD. Această zonă desemnează natura şi structura datelor destinate caracteristicilor descrise în ⟨*alte informaţii*⟩ din entitatea de legătura L ;

— ⟨*tip operaţie*⟩ : identificatorul zonei din linia de intrare care va conţine codul operaţiei de efectuat (A — actualizare ; S — ştergere).

Formalizarea descrierii identificatorilor

Pentru descrierea regulilor de producere a simbolurilor nonterminale specificate în paragraful anterior facem precizarea că valorile scrise cu majuscule sînt considerate valori implicite iar valorile date ca simbol non-terminal vor fi substituite de utilizator conform necesităţilor sale. La apelul generatorului de programe aceste simboluri

non-terminale vor constitui parametri de apel. În funcţie de atribuirea unei valori sau nu generatorul va lua în considerare acea valoare sau, respectiv, valoarea implicită.

Formalizarea descrierii identificatorilor caracteristicilor definite în entităţi de tip nomenclator apare astfel :

— ⟨*cod*⟩ : : = COD/⟨*code*⟩

⟨*code*⟩ : identificator ales de proiectant ;

$$— \langle denumire \rangle ::= \begin{cases} \text{DEN1, DEN2, \ldots, DENi} \\ \langle rdene \rangle 1,\ \langle rdene \rangle 2, \ldots,\ \langle rdene \rangle i \end{cases}$$

unde :

- Valoarea lui i depinde de modul în care suma lungimilor cîmpurilor rezervate prin caracteristicile desemnate de identificatori este mai mare sau egală cu numărul de caractere din denumire ;
- ⟨*rdene*⟩ : rădăcina denumire din entitate este un identificator ales de utilizator prin care se specifică rădăcina comună din care se formează identificatorii cîmpurilor ⟨*denumire*⟩ din toate entităţile de tip nomenclator specificate la un apel al generatorului de programe ;

— ⟨*identificator inel*⟩ : : =
⟨*indicativ entitate în care este definită caracteristica*⟩ —
⟨*indicativ entitate cu care se află în relaţia „m—n"*⟩

Celelalte caracteristici de tip inel pot fi descrise cu această regulă sau conform dorinţei proiectantului.

⟨*indicative entitate...*⟩ : se formează ca un cod mnemonic din numele entităţi, specificate, astfel :

1) dacă numele entităţii este format dintr-un singur cuvînt, va fi format din primele trei caractere ale acestuia ;
2) dacă numele entităţii este compus din :
   — două cuvinte : primele două caractere din primul cuvînt şi primul caracter care urmează după „—" ;
   — trei sau mai multe cuvinte : primul caracter din primul cuvînt, primul caracter din cel de al doilea cuvînt şi primul caracter din cel de al treilea cuvînt ;
3) dacă numele entităţii este format din 1—3 litere va fi preluat în întregime ;

— ⟨*referinţe*⟩ : : = indicativ entitate care referă ⟨*indicativ entitate referinţă*⟩ — ⟨*indicativ entitate referinţă*⟩

Regulile de producere pentru acest tip de formalizare sînt prezentate, sub formă de automate, în figura 8.20.

Formalizarea descrierii identificatorilor caracteristicilor definite în formalul asociat unei entităţi de tip nomenclator apare astfel :

— ⟨*ident*⟩ : : = IDENT/⟨*ident*⟩

Va conţine, la exploatare, identificatorul machetei de prelucrare. Acest identificator, în varianta „standard" propusă, este identic cu indicativul entităţii specificate ;

— ⟨*cod*⟩ : : = COD — ⟨*indicativ entitate*⟩ ⟨*codf*⟩ — ⟨*identificator entitate*⟩

⟨*codf*⟩ : rădăcina din care se formează identificatorul caracteristicii de tip ⟨*cod*⟩ din fiecare formal asociat entităţilor specificate la un apel ;

⟨*den* 1⟩, ⟨*den* 2⟩, — ⋮, ⟨*den i*⟩ : : = { {DEN1/DEN2.../DENi/} — ⟨*indicativ entitate*⟩ ; {⟨*rdenf*⟩1/⟨*rdenf*⟩2/.../⟨*rdenf*⟩i} — ⟨*indicativ entitate*⟩ }

⟨*rdenf*⟩ : *rădăcina denumire din formal* ;

⟨*ref* 1⟩, ⟨*ref* 2⟩, — ⋮, ⟨*ref i*⟩ : : = { ⟨*indicativ entitate*⟩ — {REF1/REF2/.../REFi} ; ⟨indicativ entitate⟩ — {⟨*rref*⟩1/⟨*rref*⟩2/.../⟨*rref*⟩i } }

⟨*rref*⟩ : rădăcina identificatorului zonei destinate recepţionării datelor necesare încărcării *referinţei*.

Regulile de producere a acestui tip de caracteristici sînt prezentate, sub formă de automate, în figura 8.21.

Formalizarea descrierii identificatorilor caracteristicilor definite în entităţile de legătură apare astfel :

— ⟨*identificator enti. leg.*⟩ : : = ⟨*identificator enti.* X⟩ — ⟨*identificator enti.* Y⟩/
⟨*identificator enti.* X⟩ — ⟨*indicativ enti.* Y⟩/
⟨*indicativ enti.* X⟩ — ⟨*identificator enti.* Y⟩
— ⟨*referire* Xy⟩ : : = ⟨*indicativ enti.* X⟩ ⟨*indicativ enti. y*⟩ ⟨*indicativ enti.* x⟩
— ⟨*cod entitate* x⟩ : : = COD — ⟨*indicativ enti.* x⟩/
⟨*rcode*⟩ — ⟨*indicativ enti.* x⟩

Automatele acestor reguli de producţie sînt prezentate în figura 8.22.

Formalizarea descrierii identificatorilor caracteristicilor definite în formalul asociat unei entităţi de legătură apare astfel :

— ⟨*cod macheta*⟩ : : = IDENT/⟨*ident*⟩

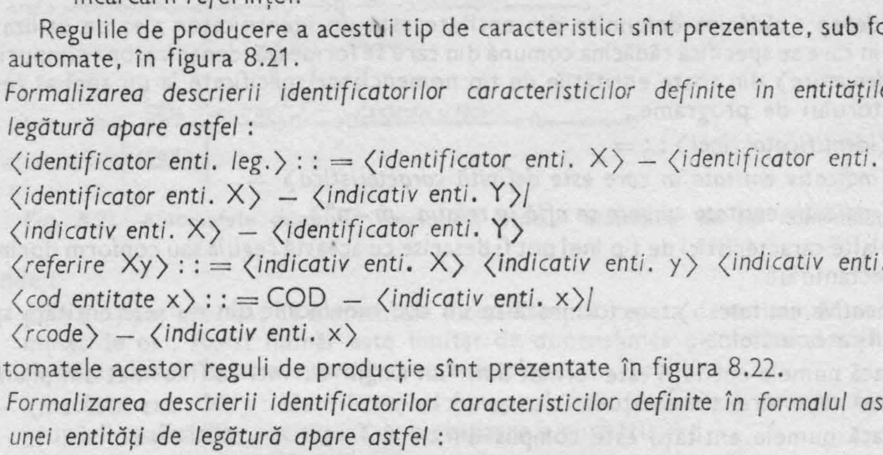


Fig. 8.22. Automatele descrierii entităţilor de tip „LEGĂTURA"

⟨*ident*⟩ : va conține la momentul prelucrării (varianta standard) codul machetei (formatului) construit astfel :

⟨*indicativ enti.* X⟩ ⟨*indicativ enti.* Y⟩ pentru entitățile X și Y aflate în relația „*m*–*n*".

— ⟨*ident. cod enti* X⟩ : : = COD — ⟨*indicativ enti-x*⟩/
⟨*rcodf*⟩ — ⟨*indicativ enti* x⟩

— ⟨*tip operație*⟩ : : = {A/S}/⟨*tipop*⟩, unde :

A : creere — adăugare relații ;

S : ștergere relații ;

— ⟨*info. enti. leg.*⟩ : nu există restricții asupra modului de definire a identificatorilor ;

{⟨*ident. cod enti.* x⟩ [⟨*infor. enti. leg.*⟩]} pot fi repetitive ;

În acest caz se va utiliza un formal repetitiv imbricat pentru care avem :

— ⟨*nr. max. formal*⟩ : numărul maxim de repetări (definit de proiectant) ;

— ⟨*ident. form. repetitiv*⟩ : : = REPFORM/⟨*reform*⟩, identificatorul formalului repetitiv definit.

Automatele asociate acestor formalizări sînt prezentate în figura 8.23.

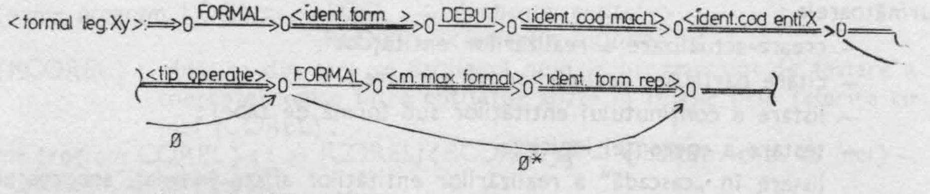


Fig. 8.23. Automatele descrierii formalului pentru entități de tip „LEGĂTURA"

Principiul funcţionării produsului de generare automată

Produsul încorporează în logica sa internă formalizarea generală a descrierii entităţilor şi a caracteristicilor de tip formal asociate („batch processing" sau conversaţional cu format). Bazat pe această formalizare şi parametrii specificaţi la apel, produsul realizează următoarele operaţii:

— validarea parametrilor de apel;
— apelul prototipului de generare din fişierul /biblioteca în care este stocat;
— analiza parametrilor de intrare şi generarea, în funcţie de valorile acestora, unui fişier (al cărui nume şi spaţiu rezervat este specificat de utilizator) în formatul cerut de intrarea standard a sistemului.

Acest fişier va conţine codul sursă al unor macroinstrucţiuni sau programe precompilate specifice gamei de operaţii cerute pentru fiecare entitate analizată. Codul sursă al fiecărei unităţi de program este precedat de apeluri, conform limbajului de comandă al implementării SOCRATE, la macrogenerator.

Acest cod sursă astfel obţinut poate fi catalogat, sub controlul SGBD-SOCRATE, în spaţiul bazei (bazelor) de date specificată(e) de utilizator. Înaintea efectuării operaţiei de catalogare utilizatorul poate interveni, eventual, pentru modificarea acestui cod sursă (prin utilizarea, de exemplu, a funcţiunilor editorului de texte al sistemului, a editorului utilizatorului bibliotecar etc.).

După catalogare, în spaţiul unei baze de date, programele pot fi utilizate pentru realizarea operaţiei (operaţiilor) dorite.

Printre operaţiile care pot fi realizate de programele generate enumerăm următoarele:

— creare-actualizare a realizărilor entităţilor;
— citare parţială a realizărilor de entitate;
— listare a conţinutului entităţilor sub formă de tabel;
— testare a coerenţei logice;
— listare în „cascadă" a realizărilor entităţilor aflate în relaţii arborescente (relaţii ierarhice);
— testarea coerenţei logice a entităţilor de tip legătură etc.

Pentru aceste operaţii enumerate prezentăm în anexa 4, principiul de funcţionare al programelor respective.

Generatorul de programe primeşte la intrare un set de parametri, majoritatea opţionali dacă sînt respectate formalizările prezentate anterior, astfel:

⟨*parametri-de-apel*⟩ : : = ⟨ENTI⟩ [,⟨TIPREF1⟩]...[,⟨TIPREFi⟩] [,⟨TIPENTI⟩]
[,⟨REDENE⟩] [,⟨RDENF⟩] [,⟨CODE⟩] [,⟨RREF⟩]
[,⟨RCODF⟩] [,⟨CODF⟩] [,⟨IDENT⟩] [,⟨FORMAL⟩]
[,⟨REPFORM⟩] [,⟨RPROG⟩] [,⟨RLIST⟩] [,⟨RLOG⟩]
[,⟨RCOREL⟩] [,⟨TIPREL⟩]

— ⟨ENTI⟩ : :='⟨*nume entitate*⟩', '⟨*tip entitate*⟩'/
'⟨*nume entitate*⟩', '⟨*tip entitate*⟩', '⟨*nume enti. ref*⟩'/
'⟨*nume entitate*⟩', '⟨*tip entitate*⟩', ⟨ENTI⟩

⟨*nume enti. ref*⟩ : : = ⟨*nume enti. ref*$_1$⟩/⟨*nume enti. ref*$_1$⟩, ⟨*nume enti ref*$_2$⟩/⟨*nume enti. ref.*⟩, ⟨*nume enti. ref*$_i$⟩

⟨ENTI⟩: poate avea maxim 64 elemente la un apel;

⟨*nume entitate*⟩: numele entităţii pentru care se generează programul (programele).

Acest nume este cel definit de ABD la descrierea structurii conceptuale a bazei de date ;

— ⟨*tip entitate*⟩ : tipul modelului de descriere în care se încadrează entitatea; ⟨*nume enti. refi*⟩ : în cazul în care între entităţi există relaţii de tip referire simplă sau cu inel se vor specifica numele entităţii către care se face referirea ;

— {⟨TIPREF1⟩/⟨TIPREF2⟩/.. /⟨TIPREFi⟩} : tipul caracteristicii cheie a realizării entităţii către care se face referirea i.

⟨TIPREFi⟩ : : = NUM/ALF/VID

- NUM : numeric ;
- ALF : alfanumeric ;
- VID : număr de realizare.

— ⟨TIPENTI⟩ : : = NUM/ALF/VID, tipul caracteristicii cheie primară a entităţii ;

— ⟨RPROG⟩ : rădăcina din care se formează numele programului pentru creere-actualizare-modificare realizări (CAM) ;

Numele programului de tip „CAM" se formează astfel :

⟨*nume program CAM*⟩ : : = CAM — ⟨*indicativ entitate*⟩/
⟨RPROG⟩ — ⟨*indicativ entitate*⟩

— ⟨RLIST⟩ : rădăcina din care se formează numele programului de listare a realizărilor entităţii ;

⟨*nume program listare*⟩ : : = LIST — ⟨*indicativ entitate*⟩/
⟨RLIST⟩ — ⟨*indicativ entitate*⟩

— ⟨RLOG⟩ : rădăcina din care se formează numele programului de testare a coerenţei logice (LOG) a entităţii ;

⟨*nume program* LOG⟩ : : = LOG — ⟨*indicativ entitate*⟩
RLOG — ⟨*indicativ entitate*⟩

— ⟨RCOREL⟩ : rădăcina din care se formează numele programului de testare a coerenţei logice între entităţile aflate în relaţie prin referire cu inel (COREL) ;

⟨*nume program* COREL⟩ : : = {COREL/⟨RCOREL⟩} — ⟨*indicativ enti. cu inel*⟩ — ⟨*indicativ enti. cu referire*⟩

— ⟨FORMAL⟩ : : = MACHETA/⟨*identificator formal*⟩ : numele caracteristicii de tip formal în care este descris formatul datelor de intrare ;

— ⟨TIPREL⟩ : : = BTCH/CONV/FORM — tipul modului de prelucrare ales pentru programele CAM (Restul programelor funcţionează implicit, în aceeaşi formă, şi aceeaşi formă, şi în conversaţional şi în „batch processing") ;

— BTCH : „batch processing". Aceste programe sînt construite şi pentru a funcţiona de tipul FORM. Funcţionarea lor în conversaţional este identică cu cea a unui program de tip FORM numai dacă lungimea liniei de intrare a datelor nu depăşeşte lungimea liniei terminalului conversaţional ;

— CONV : prelucrare conversaţională gen „întrebare-răspuns" ;

— FORM : prelucrare conversaţională cu ecran (linie) formatat(ă).

Ceilalţi parametri din modelul de apel au aceeaşi semnificaţie cu cea specificată la definirea regulilor de formalizare.

În anexa 5 este prezentat un exemplu de apel al unui prototip de generare clasat de autori în generaţia 1-a versiunea 2-a.

Ieşirile produsului sînt reprezentate de codul sursă al programelor în formatul de prezentare al intrării standard.

Caracteristic pentru acest tip de produs este faptul că el funcţionează pe „*clase de modele*". Prin „clasă de modele" înţelegem faptul că produsul este capabil să gene-

reze programe de tipul anunţat pentru diverse modele de descriere dar care au anumite trăsături comune (acelaşi tip de cod al entităţii şi entităţilor referire, aceeaşi rădăcină pentru anumiţi identificatori etc.).

Acest lucru nu diminuează rezultatele obţinute : pentru fiecare „clasă de modele" se va realiza unul sau mai multe apeluri, fiecare apel generînd un fişier. Aceste fişiere pot fi concatenate eventual, cu ajutorul funcţiunilor sistemului de operare gazdă.

Mai mult concepţia unitară a acestui gen de programe face posibilă monitorizarea lor, ceea ce aduce imense avantaje la exploatarea curentă. În acest sens programele generate pot fi considerate ca un „mediu de programare funcţională".

Dacă considerăm că programele generate sînt de tip macroinstrucţiune atunci fişierul generat are structura :

⟨cerere de conectare la baza de date specificată de utilizator⟩

⟨apel macrogenerator⟩
:SUPEXP *⟨nume program⟩*?
⟨apel macrogenerator⟩
:DEFMAC *⟨nume program⟩*
:EXP
⟨model expansiune⟩
:FDED ?

} această secvenţă se repetă
pentru fiecare program cerut la apel

⟨cerere de deconectare de la baza de date⟩

Proiectarea structurii bazelor de date în vederea exploatării simultane

Acest paragraf este destinat tratării modului în care se poate realiza integrarea bazelor de date aflate în exploatare şi modului de proiectare a bazelor de date noi în vederea exploatării simultane a acestora. Exploatarea bazelor de date se va efectua local, caz în care aceste baze sînt considerate distribuite local, sau la distanţă prin intermediul facilităţilor oferite de RENAC, caz în care bazele de date sînt distribuite la distanţă. Exploatarea „distribuită" a bazelor de date se va efectua, în condiţiile în care versiunile disponibile ale SGBD-SOCRATE nu sînt de tip distribuit, prin intermediul interfeţei cu limbaje evoluate.

Pentru a avea o bază comună de discuţie vom considera că în structura bazei de date entităţile se încadrează în următoarele tipuri : *⟨entităţi dicţionar⟩*, *⟨entităţi documentare⟩*, *⟨entităţi utilitare⟩* şi *⟨entităţi nucleu⟩*. Vom trata în continuare implicaţiile integrării bazelor de date, diferenţiat, în funcţie de starea în care se află acestea :

1) în exploatare ;

2) în proiectare.

Integrarea bazelor de date aflate în exploatare

Acest tip de integrare poate fi încadrat în unul din următoarele cazuri :

a) bazele de date sînt eterogene ca structură dar au elemente comune reprezentate în acelaşi mod (ex. : acelaşi tip de cod şi acelaşi conţinut — o bază de date

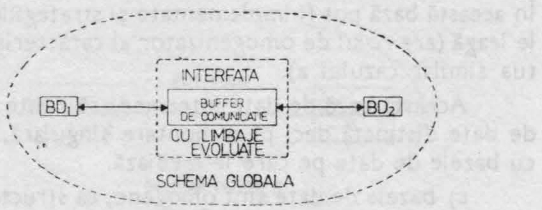


Fig. 8.24. Exploatarea simultană a bazelor de date eterogene utilizînd buffere de comunicaţie

pentru evidenţa personalului care identifică persoana prin marcă şi o bază de date a producţiei în care se face o evidenţă a operaţiilor realizate de fiecare muncitor identificat unic cu aceeaşi marcă).

Viziunea integrală asupra celor două baze de date poate fi realizată cu ajutorul interfeţei cu limbaje evoluate (COBOL, FORTRAN, ASSIRIS etc.) a SGBD-SOCRATE, utilizînd un buffer de comunicaţie între baze (fig. 8.24), căutarea în fiecare bază efectuîndu-se prin programe specifice fiecăreia dar pe baza aceluiaşi criteriu (numai din punct de vedere logic şi semantic).

Această viziune integrală este realizată la nivel logic prin structura bufferului de comunicaţie şi logica internă a programelor de exploatare a bazelor de date care utilizează acest buffer.

Dacă strategiile de căutare în bazele de date exploatate simultan nu convin din punct de vedere al timpilor de răspuns se poate introduce o bază de date intermediară (SA) a cărei structură să permită conservarea strategiilor de căutare optime în ambele sensuri (fig. 8.25).

În acest caz orice căutare în bazele de date este subordonată căutării strategiei optime de acces indicată în baza de date intermediară.

Actualizarea informaţiilor din BD1 şi BD2 incluse în strategia de acces se va efectua, în acest caz, utilizînd interfaţa şi considerînd baza de date formată din BD1-SA, BD2-SA sau BD1-SA-BD2. Baza de date care conţine strategia de acces SA va indica, în general, numerele realizărilor entităţilor din BD1, respectiv BD2, care conţin elementul de căutare. Strategiile de acces din SA vor fi astfel concepute încît să permită rezolvarea tuturor aplicaţiilor comune lui BD1 şi BD2.

b) bazele de date sînt eterogene iar elementele comune (semantic) au structuri eterogene şi conţinut eterogen (ex.: în BD1 ⟨*cod*⟩ este de tip numeric cu valori numerice iar în BD2 ⟨dcod⟩, care reprezintă semantic aceeaşi informaţie, este alfanumerică cu valori alfanumerice).

În acest caz se va construi o bază de date intermediară care va avea rolul de translator (operator de transformare împreună cu programul) între cele două baze.

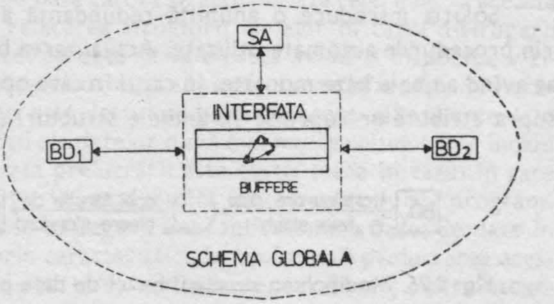


Fig. 8.25. Exploatarea simultană a bazelor de date eterogene utilizînd o bază de date intermediară care conservă strategiile optime de acces ale bazelor de date

În această bază pot fi implementate și strategiile de căutare în bazele de date pe care le leagă (are rolul de omogenizator al caracteristicii comune). Exploatarea se va efectua similar cazului a).

Această bază de date intermediară poate fi întreținută considerînd-o ca o bază de date distinctă deci prin montare singulară, sau prin montare parțială sau totală cu bazele de date pe care le mediază.

c) bazele de date sînt omogene, ca structură și semantică, în acest caz interfața va face apel la același set de programe catalogat în toate bazele de date (ex. : o bază de date pentru evidența personalului realizată pentru întreprinderi care aparțin aceleiași centrale : aplicațiile la nivel central pot fi realizate exploatînd simultan, cu ajutorul interfeței, toate bazele de date). Programul apelat pentru o bază de date sau alta poate să difere conform necesităților utilizatorului.

d) adăugarea de noi atribute la o entitate nucleu (sau la mai multe) a unei baze de date aflată în exploatare.

În acest caz una din soluții ar fi aceea a creerii unei noi baze de date în care structura cuprinde și aceste noi atribute. Datele aflate în baza de date, aflată în exploatare, vor fi extrase pe suport magnetic într-un format ales de utilizator (este preferabil ca acest format să fie de tipul „fișier slash" pentru a utiliza funcția de creare standard a SGBD-SOCRATE), iar baza nouă va fi creată utilizînd valorile salvate în acest fișier și valorile destinate atributelor adăugate (fig. 8.26).

Dacă volumul datelor este mare (bază multivolum) sau există un număr mare de relații (refere-anneau) această acțiune este foarte costisitoare (necesită recatalogarea, în noua bază, a programelor și încărcarea noii baze de date).

O soluție mai bună pentru acest caz este aceea a creerii unei noi baze de date în care vor fi definite entități nucleu cu același nume și același număr de realizări ca în baza de date veche în care vor fi definite atributele noi. În plus aceste entități vor conserva cheile de acces discriminante păstrîndu-se astfel corespondența acestor atribute la o realizare din entitatea de bază (în particular această corespondență va fi dată în mod biunivoc de numărul de realizare). În această bază nouă vor fi definite, eventual, și entități de celelalte tipuri împreună cu relațiile lor sau strategii noi de parcurgere a bazei de date. În acest caz corespondența între realizări (deci viziunea integrală a unei entități de tip uncleu) va fi realizată prin probleme scrise în LMD-SOCRATE exploatate cu ajutorul interfeței cu limbajele de nivel înalt.

Această soluție este mai puțin costisitoare decît prima și are avantajul că fiecare bază de date poate fi exploatată autonom pentru aplicațiile care solicită acest lucru sau simultan pentru cele care au o viziune integrală.

Soluția introduce o anumită redundanță a datelor, redundanță controlabilă prin procedurile automate utilizate. Actualizarea bazei de date trebuie făcută în acest caz avînd ambele baze montate, în cazul în care operațiile de actualizare se efectuează asupra atributelor cuprinse în ambele structuri (fig. 8.27).

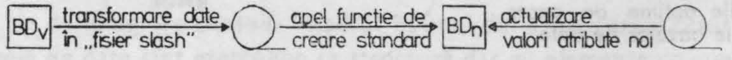
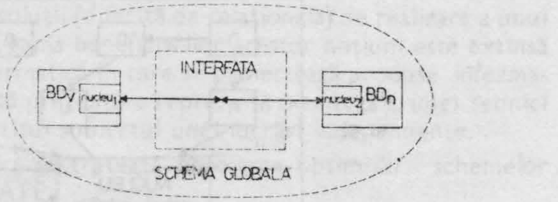


Fig. 8.26. Modificarea structurii bazei de date prin adăugarea de noi atribute

Fig. 8.27. Adăugarea de noi atribute la o bază de date aflată în exploatare utilizînd o bază de date adițională



Integrarea bazelor de date aflate în proiectare

În acest caz proiectarea trebuie să conserve structura atributelor, să le afecteze aceleași valori și să conserve strategia de acces la aceste valori. Acest lucru duce la eliminarea introducerii unei baze de date de translatare sau a unei baze de date cu strategii de acces, viziunea integrală urmînd să se efectueze cu ajutorul interfeței cu limbaje evoluate.

O problemă de distribuire locală pe care dorim să o tratăm este aceea efectuată asupra bazelor de date de mari dimensiuni, cu date cu structură omogenă, dar care nu necesită prezența tuturor datelor pentru toate aplicațiile funcționale (de exemplu o bază de date pentru evidența personalului nu necesită prezența personalului TESA și MUNCITORI pentru toate aplicațiile funcționale, deci poate fi descompusă funcțional).

În acest caz una din soluții este aceea a creerii unei baze de date care să conțină ⟨*entități nomenclator*⟩, ⟨*entități utilitare*⟩ și relațiile dintre acestea pentru întreaga bază de date precum și strategiile de acces pentru toate bazele de date (în general caracteristici inverse). În celelalte baze de date vor fi înregistrate datele care privesc strict acel domeniu (⟨*entități nomenclator*⟩, ⟨*entități utilitare*⟩, ⟨*entități documentare*⟩ și ⟨*entități nucleu*⟩ împreună cu relațiile lor). Entitățile nucleu din toate bazele de date vor conține în mod obligatoriu atributul (atributele) ale cărui valori constituie criteriul de partajare. Exploatarea se va efectua pentru aplicațiile specifice fiecărei baze de date menținînd numai acea bază de date iar pentru aplicațiile comune se va efectua avînd o schemă globală logică dată prin structura bufferelor și programelor precompilate utilizate, prin intermediul interfeței, de programele de exploatare.

O altă metodă constă în conservarea în una din bazele de date a ⟨*entităților documentare*⟩, ⟨*entităților utilitare*⟩, ⟨*entităților nomenclatoare*⟩ și a relațiilor lor și în conservarea în ⟨*entitățile nucleu*⟩ numai a cheilor de acces direct, a referirilor și inelelor (fig. 8.28). În cea de-a doua bază de date, vor fi definite celelalte atribute (conservînd cheile de acces direct) și eventual dublurile în ⟨cod⟩ ale referirilor (atribuite care să conserve valorile pe baza cărora se actualizează referirile — această soluție prezintă importanță pentru refacerea structurii inelelor în cazul distrugerii coerenței acestora). Actualizarea bazei de date se va efectua avînd la dispoziție viziunea integrală a bazei de date, exploatarea urmînd să se efectueze în funcție de necesitățile aplicațiilor funcționale. Soluția este destul de bună deoarece la bazele de date mari în cazul în care se execută selecții ale datelor după mai multe criterii care includ și atribute secvențiale (AYANT) durata prelucrării este costisitoare în cazul în care întrebările și răspunsurile sînt cerute în mod direct în cadrul aceluiași program. În acest caz este preferabilă efectuarea extragerii unei subscheme a bazei de date în bandă sau disc (subschemă definită prin caracteristici de tip formal) prelucrarea acesteia prin programe clasice (operații de sortare, centralizare, calcule etc.) și reîntoar-

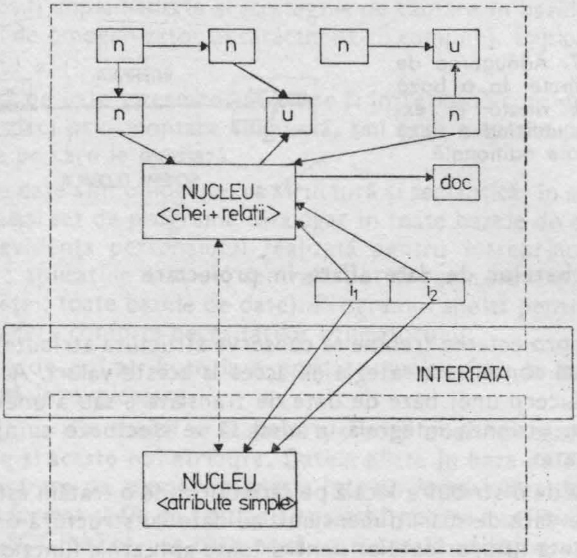


Fig. 8.28. Partajarea bazei de date prin separarea atributelor simple ale entităţilor nucleu

cerea la baza de date pentru editarea semnificaţiei finale a indicatorilor calculaţi sau a codurilor utilizate (în general valori aflate în caracteristici definite în entităţile de tip dicţionar (nomenclator).

Utilizarea facilităţilor puse la dispoziţie de RENAC se poate realiza înglobînd în rădăcina programului de interfaţă modulele de acces la reţea.

Pentru implementările FELIX C — se va modifica comanda .TREE, pentru obţinerea schemei de reacoperire a segmentelor, astfel :

•TREE rprg $^{0}/_{00}$ (MARC ... restul argumentelor din comanda standard pentru interfaţa cu limbaje evoluate).
La exploatarea simultană a mai multor baze de date trebuie poziţionat corespunzător indicatorul UN al comenzi .OPTION (UN : nr. baze, unde nr. baze este numărul de baze de date simultan active).

O bază de date Mini poate comunica cu o bază de date FELIX utilizînd dispozitivele puse la dispoziţie de către RENAC tot prin intermediul interfeţei. Pentru această implementare este posibilă comunicarea între baze de date aflate în maxim 11 noduri simultan prin lansarea din cadrul unui program, considerat „master" (lansat în nodul care devine coordonator, pentru această aplicaţie) a unor programe care îşi desfăşoară activitatea pe baze de date aflate în alte noduri (logice sau fizice) considerate „slave"

Concluzii

Acest capitol a fost destinat tratării problemelor de optimizare a structurilor şi programelor de aplicaţie ale unei baze de date. În prima parte este prezentată o metodă de obţinere a unei structuri optime din punct de vedere al eliminării anoma-

liilor de exploatare comune oricărei soluţii (diferită de relaţională) de realizare a unui sistem informatic. În acest context gama beneficiarilor acestor noţiuni este extinsă pe întreg domeniul de activitate informatică în care se proiectează produse informatice care lucrează cu „fişiere". Această prezentare reprezintă o sinteză a unei tehnici susţinute de autori, care poate constitui subiectul unei lucrări independente.

În contextul acestei prezentări se tratează problema optimizării schemelor conceptuale descrise cu LDD-SOCRATE.

Optimizarea este tratată analizînd modul de gestiune a informaţiilor în spaţiile alocate bazei de date. Cu toate că modul de gestiune are o amprentă specific SOCRATE conţine principiile de manipulare şi gestiune ale unei memorii paginate, care pot fi utile, la nivel de cunoaştere generală, oricărui informatician.

Deşi, în ansamblu, capitolul tratează noţiuni independente de implementare, am tratat într-un paragraf modul de analiză a imaginii memoriei virtuale pentru implementarea V 1.5 din următoarele raţiuni :

— această versiune este „îngheţată" la acest mod de interpretare ;

— această versiune este disponibilă pentru testare şi utilizare în mod gratuit ;

— principiile conţinute în această prezentare sînt valabile pe celelalte implementări.

Din aceste considerente recomandăm să fie studiată indiferent care va fi implementarea pentru care optează utilizatorul.

În acest capitol este prezentată o metodă de optimizare pusă la punct în anul 1978 şi testată cu foarte bune rezultate pe mai multe structuri de baze de date aflate în prezent în exploatare. Atragem atenţia că această metodă trebuie utilizată indiferent dacă implementarea oferă opţiunea realizării automate, la momentul definirii, a alinierii primei realizări de entitate la frontieră de subpagină.

În principiu, algoritmii de optimizare prezentaţi sînt destinaţi ABD.

O altă problemă de optimizare este aceea destinată înzestrării bazei de date cu programe de „serviciu" pentru diverse operaţii aflate în sarcina ABD. Pentru fiecare tip de program se dă un algoritm generalizat din care se pot obţine, prin efectuarea substituţiilor elementelor formale încorporate, programele specifice, rezolvării acestui gen de probleme, unei baze de date particulare.

Aceste tehnici de construire au fost prezentate la optimizare deoarece codul programului rezultat, după substituţie, este compact, performant şi corect iar utilizatorul nu mai trebuie să-şi pună problema obţinerii acestui gen de programe.

O parte importantă a capitolului este destinată formalizării structurilor bazelor de date şi a descrierii identificatorilor cu scopul principal al oferirii unui mediu algoritmic pentru generarea automată a programelor. Pentru generarea automată este oferită o metodă care cere un efort minim de programare din partea utilizatorilor.

Noţiunile introduse în acest capitol pot fi utilizate de ABD pentru a defini standardele sale impuse pentru o bază de date (conform atribuţiei sale de a „forţa" standardizarea). Chiar dacă ABD nu doreşte să genereze automat programele sale formalizarea poate fi utilizată, pe de o parte pentru a avea o regulă de stil riguros formulată la definirea structurii, iar pe de altă parte pentru a beneficia de posibilitatea construirii unor programe generalizate adaptabile parametric, semantic şi contextual, la o execuţie particulară. Regula de formalizare a fost utilizată parţial la defi-

nirea structurii conceptuale a bazei de date BDDPERS, prezentată ca exemplu în capitolul 3. Utilizarea parțială a formalizării a fost dictată de rațiunea de a nu deforma imaginea, despre descrierea structurii utilizatorului care ia contact cu SGBD-SOCRATE prin această lucrare. Și în acest caz enormele avantaje ale formalizării pot fi observate în modul de construire a exemplelor din capitolul 5, la prezentarea macrogeneratorului.

Ultima parte a capitolului este destinată modului de proiectare a structurii bazelor de date în vederea beneficierii de avantajele prelucrării distribuite, locale sau la distanță, în contextul unei implementări nespecializate.

9. DEZVOLTĂRI SOCRATE ȘI ALTE SGBD-uri

Introducere

Ultima versiune SOCRATE adusă la noi în țară a fost versiunea V 1.5. Asupra acestei versiuni, diferite centre din țară, au adus o serie de dezvoltări și îmbunătățiri (ICSIT-TCI a realizat versiunea V 1.6R, CTCE Constanța a realizat versiuni pe minicalculator, introducîndu-se și unele facilități din V 1.6 și V 1.7, Centrul de Calcul CSP au introdus numerele zecimale etc.).

Francezii, pînă în prezent, au adus o serie de dezvoltări, pe care le vom prezenta în paragrafele următoare. Vom insista mai mult asupra sistemului CLIO deoarece este versiunea prezentă și deoarece este mai puțin cunoscută la noi în țară, în comparație cu V 1.6, V 1.7 și V 1.8.

O altă direcție a constituit-o dezvoltarea unor subseturi ale SGBD SOCRATE, pentru a răspunde anumitor genuri de aplicații. Asupra acestei direcții nu vom insista. Cei ce doresc informații în plus pot studia cele trei teze de doctorat despre sistemul SOMINE [MATY], [GAILLARD], [SAYETTAT], existente în țară.

O direcție mai importantă este cea a dezvoltării SGBD SOCRATE pentru gestiunea distribuită a datelor, direcție ce va fi abordată în acest capitol.

Dezvoltări ECA AUTOMATION

ECA Automation a propus și a dezvoltat SGBD SOCRATE în trei versiuni noi: V 1.6, V 1.7, V 1.8; pe care le vom trata în continuare, atît pentru a servi ca sursă de informare pentru cei ce doresc să dezvolte SGBD SOCRATE, cît și pentru a cunoaște modul în care se încadrează sistemul pe care îl deține fiecare, în raport cu aceste dezvoltări.

Versiunea V1.6

Versiunea V1.6 a fost propusă în 1976 [V1.6.76] şi s-a revenit asupra ei în 1977 [V1.6.77] şi în 1978 [V1.6.78]. Dezvoltările şi funcţiile noi pe care le-a adus versiunea V1.6 sînt următoarele :

— memoria asociativă şi segmentarea,
— apelul direct al programelor compilate,
— jurnal rapid pe disc magnetic,
— posibilitatea de a introduce programe utilizator în baza de baze,
— adăugarea spaţiilor virtuale „structură" şi „lucru",
— datarea mesajelor către terminalul de control,
— emiterea unui mesaj către un terminal,
— creşterea numărului de terminale,
— declanşarea unui punct de reluare în funcţie de ceas,
— posibilitatea de a defini drepturi de acces la actualizare,
— condiţionarea efectului comenzii LOGOUT,
— zona de comunicaţii în memoria principală,
— contoare de evaluare în modul de lucru conversaţional,
— fişiere de date secvenţiale pe disc,
— comandă explicită de închidere a fişierelor,
— editare formatată pe caracteristica FORMAL,
— definirea de numere zecimale,
— variabile zecimale W_i,
— teste pe caracteristicile FORMAL.

În continuare vom descrie pe scurt fiecare din aceste dezvoltări.

Memoria asociativă. Se rezervă într-o zonă de memorie un număr de cuvinte, în care vor fi plasate informaţiile necesare pentru gestiunea programelor indicate de un utilizator. Se evită în acest fel apelurile succesive ale acestor programe şi căutarea sistematică în biblioteca de programe compilate.

Zona de segmentare. Cînd anumite programe, care au fost indicate pentru a figura în memoria asociativă, au o frecvenţă de execuţie mare şi talia lor permite aducerea lor completă în memoria principală, utilizatorul are posibilitatea să definească o zonă, numită de segmentare. Astfel, la primul apel programul este adus în memoria principală (zona de segmentare), unde rămîne rezident pentru execuţiile ulterioare. Mai mult, utilizatorul poate defini şi un arbore de segmentare. Programul astfel segmentat este încărcat, într-o manieră continuă într-un fişier numit LOAD-GO. La încărcarea în memorie programul este luat din fişierul LOAD-GO.

Apelul direct al programelor compilate. V1.6 autorizează apelul direct al programelor compilate, cu parametrii opţionali. Parametrii, atunci cînd există, sînt pasaţi prin intermediul variabilelor Z_i. Jurnalul rapid pe disc. S-a adăugat, la jurnalele pe bandă, un jurnal rapid pe disc magnetic. Acest jurnal conţine toate zonele din baza de date, modificate de la ultimul punct de reluare. În caz de incident, pentru a reveni la ultimul punct de reluare se utilizează acest jurnal. Jurnalele pe bandă magnetică sînt utilizate în continuare, deoarece pot fi folosite pentru a se reveni la un punct de control mai vechi.

Introducerea de programe utilizator în baza de baze. Cu ajutorul SGBD SOCRATE pot fi gestionate, pe acelaşi calculator, mai multe baze de date. Programele de aplicaţii, scrise în limbajul de manipulare a datelor, sînt memorate în spaţiile de program ale fiecărei baze de date. S-a observat că există multe programe ce pot fi folosite, fără modificări, de către utilizatorii mai multor baze de date. Pentru a nu mai memora acelaşi program pentru fiecare bază de date, V1.6 permite memorarea acestor programe generale în baza de baze, de unde pot fi folosite pentru toate bazele de date.

Adăugarea spaţiilor virtuale „structură" şi „lucru". O bază de date utilizează trei tipuri de spaţii virtuale : fişier, dicţionar şi program. Spaţiul program conţine programele bazei de date în format binar executabil. Spaţiul fişier conţine informaţii privind baza de date, forma codificată a schemei bazei de date, sursele programelor, sursele macroinstrucţiunilor şi forma codificată a macroinstrucţiunilor (este deci un spaţiu cu conţinut neuniform). Spaţiul dicţionar conţine dicţionarul caracteristicilor de tip cheie, numele programelor şi al macroinstrucţiunilor. Pentru a se evita coliziunile de nume, precum şi preluarea de spaţiu de lucru din baza de baze pentru lucrul conversaţional, V1.6 adaugă încă două noi spaţii virtuale pentru fiecare bază de date :

– spaţiul de structură care să conţină toate informaţiile referitoare la schema bazei de date, a programelor şi a macroinstrucţiunilor ;
– spaţiul de lucru care va conţine zonele de lucru ale diverselor procesoare.

Datarea mesajelor către terminalul de control s-a introdus pentru a facilita eventualele căutări privind instanţele în care s-au produs anumite evenimente.

Emiterea unui mesaj către un terminal citat a fost introdusă ca o extensie la posibilităţile existente în V1.5 (transmiterea unui mesaj la toate terminalele sau la toate terminalele aceleiaşi linii).

Creşterea numărului de terminale. La V1.5 numărul maxim de terminale suportate era limitat la valorile :

– pentru modul de transmitere caracter – 48 de terminale,
– pentru modul de transmitere mesaj – 5 linii a 10 terminale fiecare.

La versiunea V1.6 aceste numere au fost ridicate la valorile : 145 de terminale şi respectiv 26 de linii a 10 terminale fiecare.

Declanşarea unui punct de reluare în funcţie de ceas. Calculatorul trebuie să aibă opţiunea de ceas şi să permită utilizarea macroinstrucţiunii RTIMER.

Definirea drepturilor de acces la actualizare. V1.6 oferă posibilitatea definirii drepturilor de acces la actualizare prin parametrii în ordinul UTI. Orice tentativă de scriere în baza de date se traduce, pentru un utilizator neautorizat, printr-o întrerupere a lucrului :

– sfîrşit anormal la interfaţa cu limbajul de programare,
– trecere la următorul ordin %, în modul batch,
– retur la nivelul $, în modul de lucru conversaţional.

Condiţionarea efectului comenzii LOGOUT. La V1.5 orice comandă LOGOUT se traduce, în modul de lucru conversaţional, într-o cerere de punct de control, indiferent dacă utilizatorul a făcut sau nu o actualizare în baza de date. La V1.6, comanda LOGOUT provoacă lansarea unui punct de control numai dacă utilizatorul are dreptul de a efectua actualizări.

Zonă de comunicaţii în memoria principală la care au acces mai mulţi utilizatori. Lungimea acestei zone este definită prin ordinul OPTION. Utilizarea acestei zone se face în acelaşi fel cu zona descrisă prin FORMAL.

Contoare de evaluare în modul de lucru conversaţional. V1.6 a introdus un anumit număr de contoare pentru colecţionarea anumitor date statistice. Aceste contoare pot fi listate cu ajutorul comenzii operator STAT sau prin comanda TERM S.

Fişiere de date secvenţiale pe disc. V1.6 ridică restricţia existentă fişiere de manevră doar pe bandă magnetică – extinzînd ordinul ATTACH şi pentru fişiere secvenţiale pe disc.

Comandă explicită de închidere a fișierelor. La V1.5 nu există un ordin explicit de închidere a fișierelor. V1.6, prin ordinul %RELEASE, adaugă această funcție, permițînd eliberarea și, deci, reutilizarea indexului de fișier și a spațiului buffer.
Editare formatată pe caracteristica FORMAL. V1.6 ridică această restricție a lui V1.5.
Definirea de numere zecimale. V1.6 ridică restricția lui V1.5, care admitea în baza de date numai numere întregi, permițînd definirea și utilizarea :
– numerelor zecimale (caracteristica DECIMAL),
– numerelor zecimale împachetate sau despachetate (caracteristicile PACKE, DILATE în cadrul blocului FORMAL),
– numerelor în format binar (caracteristica BINAIRE din cadrul blocului FORMAL),
cu următoarea *sintaxă* :

$$\textit{identificator}\ \text{DECIMAL}\ [(]\ n_1\ [\text{V}n_2\ ]\ [)]\ \text{DE}\left[\begin{Bmatrix}+\\-\end{Bmatrix}\right] nr.\ 1\ \text{A}\ \left[\begin{Bmatrix}+\\-\end{Bmatrix}\right] nr.\ 2$$

$$\textit{identificator}\begin{Bmatrix}\text{PACKE}\\\text{DILATE}\end{Bmatrix}[(]\ [n_i]\ n_1\ [\text{V}\ n_2]\ [)]$$

$$\textit{identificator}\ \ \text{BINAIRE}\ \begin{Bmatrix}[n_i]\ nr\\([n_i]\ nr)\end{Bmatrix}$$

unde :
n_1=numărul maxim de cifre pentru partea întreagă
n_2=numărul maxim de cifre pentru partea fracționară
nr_1 1=limita inferioară a intervalului
nr_2 2=limita superioară a intervalului
n_i=poziția de început din buffer
nr=poate lua două valori : valoarea 2 (binar semicuvînt) sau valoarea 4 (binar cuvînt).

Variabile zecimale. V1.6 introduce 15 variabile, W_i, de tip zecimal cu 15 poziții pentru partea întreagă și 7 poziții pentru partea zecimală. Variabilele W_i pot fi folosite în expresii aritmetice, dar nu se permite utilizarea lor în aceeași expresie cu variabilele Y_i. Sînt admise, totuși, transferuri din variabilele Y_i în W_i și invers.
Teste pe caracteristicile FORMAL. V1.6 permite teste pe caracteristicile FORMAL, sub forma

$$\text{SI}\ \left[\begin{Bmatrix}\text{EXISTE}\\\text{PAS}\end{Bmatrix}\right]\ \text{caracteristică-din-formal ALORS...}$$

În anexele de la [V1.6.76, V1.6.77, V1.6.78] pot fi găsite structura și sintaxa noilor ordine și parametrii lor.

Versiunea V1.7

Versiunea V1.7 a fost definită în anul 1976 [V1.7.76]. Principalele elemente de noutate pe care le aduce sînt : noțiunea de „schemă externă" (substructura) și posibilitatea de a asocia mai multe spații virtuale la aceeași bază de date. În plus, au fost aduse și alte dezvoltări, mai puțin importante și anume :
- directiva BORNE SOUS-PAGE,
- rezervarea de zone nespecificate,
- optimizarea accesului prin număr de ordine al realizării,
- tratarea condițiilor de eroare la nivelul limbajului de manipulare a datelor,
- cererea STOP.

Asocierea mai multor spaţii virtuale la aceeaşi bază. Prin utilizarea unei etichete virtuale la nivelul limbajului de definire a datelor, se precizează spaţiul virtual în care vor figura valorile caracteristicilor care o urmează. Definirea mai multor etichete virtuale pentru aceeaşi structură, permite repartizarea bazei de date asociate pe mai multe spaţii virtuale. Fizic aceste spaţii virtuale se pot afla :

— în acelaşi fişier,
— în fişiere diferite.

Printre avantajele acestei tehnici menţionăm :

- montarea parţială a suporţilor fizici ai bazei de date,
- fuziunea mai multor baze de date,
- ameliorarea performanţelor printr-o repartiţie, mai bine controlată, a diverselor valori din baza de date.

Sintaxa etichetei virtuale este următoarea :

$ *etichetă ⟨listă de caracteristici* LDD⟩ ⅄*etichetă*

Lista de caracteristici constituie domeniul peste care se extinde eticheta virtuală. La definirea etichetei virtuale trebuie respectate următoarele reguli :

— $ etichetă şi ⅄ etichetă trebuie să apară la acelaşi nivel din cadrul structurii,
— o etichetă virtuală poate fi definită fie la nivelul cel mai înalt al structurii, fie imediat în interiorul unei entităţi de nivel mai înalt,
— orice apariţie a unei etichete virtuale într-o structură antrenează dezimbricarea caracteristicilor din domeniul peste care se extinde, în raport cu caracteristicile din afara domeniului.

În tabela 9.1 se prezintă un exemplu în care entităţile PERSOANA şi AUTOTURISM sînt definite în două spaţii virtuale, SPA1 şi SPA2.

Tabelul **9.1. Exemplificarea spaţiilor virtuale**

```
$ SPA1
DEBUT
    ENTITE 1000 PERSOANA
    DEBUT
    .
    .
    .
    FIN
    $SPA2
        ENTITE 5 AUTOTURISM
        DEBUT
        .
        .
        .
        FIN
    *SPA2
FIN
*SPA1
```

Spaţiul virtual SPA1
PERSOANA 1
PERSOANA 2
PERSOANA 3
.
.
.

Spaţiul virtual SPA2
AUTOTURISM 1
AUTOTURISM 2
AUTOTURISM 3
.
.
.

Fiecare spaţiu virtual, definit de o etichetă virtuală, este proiectat într-un spaţiu real a cărui dimensiune de subpagină este fixată în funcţie de repartiţia dorită a datelor. Mai mult, spaţiile reale pot fi alocate pe diferite fişiere ale sistemului de exploatare, ceea ce permite lucrul cu montare parţială a volumelor bazei de date. La conectarea la o bază de date utilizatorul precizează diferitele spaţii virtuale pe care doreşte să lucreze.

Substructura. O substructură este un subansamblu de constituienţi ai structurii bazei de date care respectă imbricarea ierarhică şi ordinea existentă în structură. Pentru caracteristicile de tip REFAIRE, ANNEAU sau INVERSE, citate într-o substructură, trebuie să existe în acea substructură şi caracteristicile cu care sînt în relaţie. Într-o substructură se pot defini drepturi de actualizare, astfel :

— pentru ENTITE, ANNEAU şi INVERSE — generare (G), suprimare (S),
— pentru celelalte tipuri de caracteristici — actualizare (M).

În substructură nu apar etichete virtuale. La conectarea la o bază de date, utilizatorul trebuie să specifice substructura pe care va lucra. Un exemplu de definire a substructurilor este prezentat în tabela 9.2.

Tabela **9.2. Definirea substructurilor**

Structura

```
DEBUT
   ENTITE 2000 CENTRALA
   DEBUT
   | NUME TEXTE (10)
   | DIR-GENERAL MOT (10)
   | ENTITE 60 INTREPRINDERE
   | DEBUT
   | | NUMEI TEXTE (15)
   | | ADRESA TEXTE (20)
   | | BUGET DE 1000 A 5000
   | | ENTITE 3000 MUNCITOR
   | | DEBUT
   | | | NUME MOT (20)
   | | | FUNCTIE MOT (10)
   | | | SEX (2 8) (MASCULIN FEMININ)
   | | | RETRIBUTIA DE 2000 A 10000
   | | | VIRSTA DE 18 A 65
   | | FIN
   | FIN
   | ENTITE 7 COMPARTIMENT
   | DEBUT
   | | NR-PERS-MUNCITOR DE 10 A 50
   | | FOND-RETRIBUIRE DE 100000 A 5000000
   | FIN
   FIN
FIN
```

Substructura SUB 1

```
DEBUT
| ENTITE CENTRALA (G)
| DEBUT
| | NUME (M)
| | ENTITE COMPARTIMENT (S)
| | DEBUT
| | | NR-PERS-MUNCITOR (M)
| | | FOND-RETRIBUIRE (M)
| | FIN
| FIN
FIN
```

Substructura SUB 2

```
DEBUT
| ENTITE CENTRALA (G)
| DEBUT
| | NUME
| | DIR-GENERAL (M)
| | ENTITE INTREPRINDERE
| | DEBUT
| | | ADRESA
| | | BUGET (M)
| | FIN
| FIN
FIN
```

Directiva BORNE SOUS-PAGE. Această directivă a fost introdusă pentru a forţa alinierea la limită de subpagină. Acest lucru este util pentru a optimiza gradul de umplere a spaţiului real, pe de o parte, şi pentru a micşora timpul de acces la date, pe de altă parte.

Rezervarea de zone nespecificate. Cu ajutorul caracteristicii FILLER se rezervă loc în spațiul fișierului virtual pentru modificarea sau adăugarea ulterioară de caracteristici la structura bazei de date. Caracteristicile FILLER nu trebuie să fie utilizate în limbajul de manipulare a datelor.

Tratarea condițiilor de eroare. La execuția cererilor pot să apară mai multe erori. Codul erorii este introdus de V1.7 într-un registru numit ERREUR. Acest registru poate fi testat în cadrul programelor. Acțiunile ce pot fi efectuate asupra acestui registru sînt :

$$\text{SI ERREUR} = \left\{\begin{array}{l}\text{număr-întreg}\\ Y_i\\ \text{U}\end{array}\right\} \text{ALORS...FIN}$$

$$\text{SI} \left\{\begin{array}{l}\text{EXISTE}\\ \text{PAS}\end{array}\right\} \text{ERREUR ALORS...FIN}$$

M ERREUR=U

Valoarea nedefinită (U) din registrul ERREUR specifică faptul că nu a fost detectată nici-o eroare.

Cererea STOP. Cererea STOP permite întreruperea execuției programului de aplicații SOCRATE.

Versiunea V1.8

Versiunea V1.8 [V1.8.77] a fost propusă în anul 1977. Dezvoltările pe care le-a adus V1.8 sînt următoarele :

- introducerea de valori fără validare SOCRATE,
- acces grupat,
- operații logice pe șiruri de biți.

Obiectivele versiunii V1.8 au fost minimizarea acceselor la disc și a timpului de execuție.

Introducerea de valori fără validare SOCRATE. SGBD SOCRATE efectuează o validare și o codificare în format intern la memorarea datelor, pentru caracteristicile de tip : cuvînt, listă de valori și interval numeric. La extragerea acestor date, SOCRATE le decodifică pentru a le reda valorile exploatabile.

Pentru o serie de aplicații, datele sînt, adesea, verificate înainte de a fi furnizate sistemului de gestiune a datelor. Pentru aceste aplicații s-au introdus în V1.8 noțiunile de segment și caracteristică fără validare. Utilizarea noțiunii de segment nu antrenează, pentru caracteristicile care-i aparțin, nici validare și nici codificare. Caracteristicile unui segment se numesc caracteristici fără validare și se definesc astfel : pentru caracteristica de tip cuvînt se rezervă un byte pentru lungimea caracteristicii, care va fi completat de utilizator : pentru caracteristica listă de valori se adaugă, la definire, cuvintele NON CODEE ; pentru intervalele numerice zecimale nu se schimbă nimic ; pentru caracteristicile binare se introduce NON CODEE.

Acces grupat. Există multe aplicații care solicită accesul la date grupate și nu punctual. Pentru a răspunde la astfel de cereri V1.8 a propus introducerea noțiunii de segment (o zonă din memoria centrală, unde caracteristicile se află plasate într-o manieră

Fig. 9.1. Exemplu de realizare a segmentelor

contiguă. Aceste caracteristici sînt imaginile fidele ale unuia sau ale mai multor grupuri de caracteristici contigue din baza de date. Un segment se caracterizează prin :

— toate caracteristicile sale trebuie să fie de tip NON CODEE (caracteristici fără validare),
— nu sînt permise în cadrul segmentului caracteristicile de tip bloc adresă sau cheie,
— un segment se defineşte pe o structură sau pe o substructură, dar nu face parte propriu-zis dintr-o bază de date,
— o caracteristică sau un grup de caracteristici poate să aparţină la mai multe segmente.

Se asociază un X_1 segmentului şi caracteristicile sale pot fi utilizate în limbajul de manipulare a datelor, calificate cu variabila X_1 respectivă. Nu se realizează nici-o validare sau codificare la operaţiile de scriere a caracteristicilor segmentului. Un exemplu de realizare a segmentelor este prezentat în figura 9.1.

Operaţii logice pe şiruri de biţi. Pot fi constituite şi parcurse subansambluri de date din realizările unei clase de entităţi, care răspund unui criteriu. Acest lucru este realizabil utilizînd caracteristicile şir de biţi (de exemplu, caracteristica de tip inversă). Pot fi realizate mai multe caracteristici de acest tip. Pentru a realiza căutări combinate pe aceste caracteristici sînt utilizate instrucţiunile SI...ALORS. Dar aceste operaţii sînt destul de dificil de programat şi sînt şi costisitoare ca timp de execuţie. Pentru a accelera aceste tratări, versiunea V1.8 introduce facilitatea de operaţii logice pe şiruri de biţi. Sintaxa operaţiilor logice pe şiruri de biţi este următoarea :

$$\text{M nume-1} = [\text{PAS}]\ \text{nume-2} \left[\begin{Bmatrix}\text{ET}\\ \text{OU}\end{Bmatrix}\right] [\text{PAS}]\ \text{nume-3}$$

unde :

nume-1, 2, 3 pot fi nume de entitate sau de inversă (şir de biţi de existenţă). Pentru a clarifica modul de lucru prezentăm exemplul din tabela 9.3. În şirul de biţi LUCRU sînt identificaţi toţi locuitorii din Bucureşti, care îndeplinesc funcţia de cercetător.

Tabelul 9.3. **Exemplificarea operaţiilor logice pe şiruri de biţi**

Structura datelor

```
ENTITE 2 500000 LOCUITOR
  DEBUT
   .
   .
   .
  FIN
BUCURESTEAN INVERSE TOUT LOCUITOR
CERCETATOR  INVERSE TOUT LOCUITOR
LUCRU       INVERSE TOUT LOCUITOR
```

Programul

```
M LUCRU = BUCURESTEAN ET CERCETATOR
POUR TOUT LUCRU
   .
   .
   .
FIN
```

Concluzii

Grupul ECA Automation a propus şi a implementat mai multe dezvoltări pentru SGBD SOCRATE. Mai importante ni se par următoarele dezvoltări :

— lucrul cu fişiere pe disc magnetic,
— lucrul cu numere zecimale,
— introducerea noţiunii de schemă externă,
— introducerea de programe utilizator în baza de baze,
— asocierea mai multor spaţii virtuale la aceeaşi bază de date,
— accesul grupat.

Acestea au ridicat unele restricţii majore ale sistemului şi au apropiat sistemul de gestiune SOCRATE de un anumit standard (de exemplu, în ceea ce priveşte nivelele de structurare a datelor s-a apropiat de standardul CODASYL).

Toate dezvoltările ECA Automation au fost folosite de grupul SYSECA la sistemul CLIO, de care ne vom ocupa în paragraful următor.

Dezvoltările grupului SYSECA — Sistemul CLIO

Introducere

În prezent, SGBD SOCRATE cu funcţiuni în plus, sub o altă filosofie, cu o serie de utilitare noi este comercializat sub numele de CLIO [CLIO].

CLIO este prezentat ca un mediu omogen şi complet, de dezvoltare şi exploatare a aplicaţiilor, destinat informaticienilor şi utilizatorilor finali (utilizator neinformatician care beneficiază în mod direct de aplicaţiile informaticii) ai întreprinderii. CLIO se compune, în mare, dintr-o metodă de acces, care permite gestiunea bazelor de date de tip reţea (care au la bază modelul de date reţea) şi un dicţionar de date, care permite gestiunea viziunilor relaţionale ale bazelor de date reţea.

Sistemul răspunde la două mari categorii de cerințe ale întreprinderii :

— cerințe de gestiune a datelor, care :

- pot fi identificate dinainte și, deci, pot fi programate de către informaticieni,
- sînt repetitive (fac obiectul unei exploatări repetate),
- sînt dificile (atît din punct de vedere al dezvoltării cît și al exploatării), dar sînt în număr finit ;

— cerințe de informare și de ajutor a deciziei, care :

- sînt imprevizibile, deci nu pot fi programate dinainte,
- sînt punctuale (nerepetitive sau cu o frecvență de repetare scăzută),
- sînt în număr mare, dar simple.

În figura 9.2 se prezintă componentele CLIO și categoriile de cerințe și utilizatori, satisfăcute de aceste componente.

Pentru gestiunea datelor, CLIO se bazează pe un SGBD cu dublă orientare :

— rețea — pentru necesitățile de gestiune,

— relațională — pentru necesitățile de informare.

Pentru prima categorie de necesități CLIO furnizează :

— pe planul gestionării datelor

- un instrument evoluat de gestiune a bazelor de date ierarhice și rețea,
- diferite utilitare de administrare a acestor baze de date,
- un dicționar de date, întreținut automat ;

— pe planul manipulării datelor

- interfețe cu limbajele de programare (Cobol, PL/1, Pascal, Fortran, ...),
- interfețe cu sisteme de gestiune a tranzacțiilor (CICS, TDS, ...),
- un limbaj de programare interactiv,
- un macrogenerator condițional, care permite dezvoltarea de macrolimbaje,
- un editor de rapoarte,
- o interfață editor de texte,
- o interfață de gestiune a ecranelor.

Pentru necesitățile de informare CLIO furnizează :

— pe planul gestionării datelor

- un instrument de gestiune a viziunilor relaționale ale bazelor de date rețea,
- un dicționar de date,
- o interfață cu dicționarul de date, care permite utilizatorului să completeze dicționarul cu informații specifice întreprinderii sau să facă acces la informațiile din dicționar ;

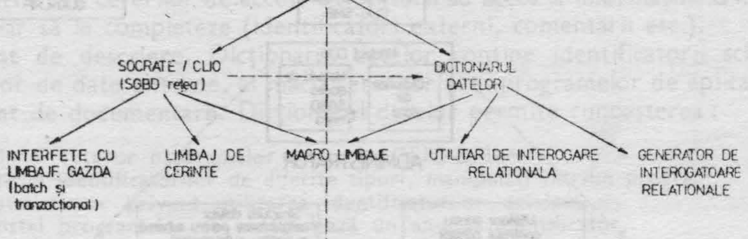


Fig. 9.2. Componentele sistemului CLIO

— pe planul manipulării datelor

- un interogator relaţional, numit şi instrument de interogare ghidată,
- macrolimbaje,
- un limbaj de manipufare a datelor,
- un editor de rapoarte,
- un gestionar de ecrane,
- un generator de interogatoare relaţionale.

Aşa cum reiese din paragraful anterior, CLIO este un produs „deschis" şi anume:
— pot fi adăugate noi funcţiuni

- îmbogăţirea limbajului de manipulare a datelor (CLIO-DML4),
- definirea altor limbaje de manipulare a datelor,
- definirea altor instrumente de interogare relaţională ;

— se permit extensii

- evoluţia schemei fizice sau modificarea implantării fizice a datelor (schimbarea de suporţi de memorare, schimbarea taliei fişierelor),
- evoluţia schemei logice (adăugări/ştergeri de entităţi, cîmpuri, relaţii, chei de acces) ;

— poate fi transferat, cu eforturi minime pe noi game de calculatoare (80% din produs este codificat într-un limbaj portabil).

În figura 9.3 se prezintă o schemă de ansamblu a componentelor sistemului CLIO. În continuare se prezintă, mai detaliat, componentele mai importante.

Nucleul CLIO

Nucleul sistemului CLIO are trei componente : CLIO-DBN — care permite definirea bazei de date şi care gestionează accesul la baza de date, CLIO-DD — dicţionarul datelor, CLIO-DBA — un ansamblu de facilităţi pentru administratorul bazei de date.

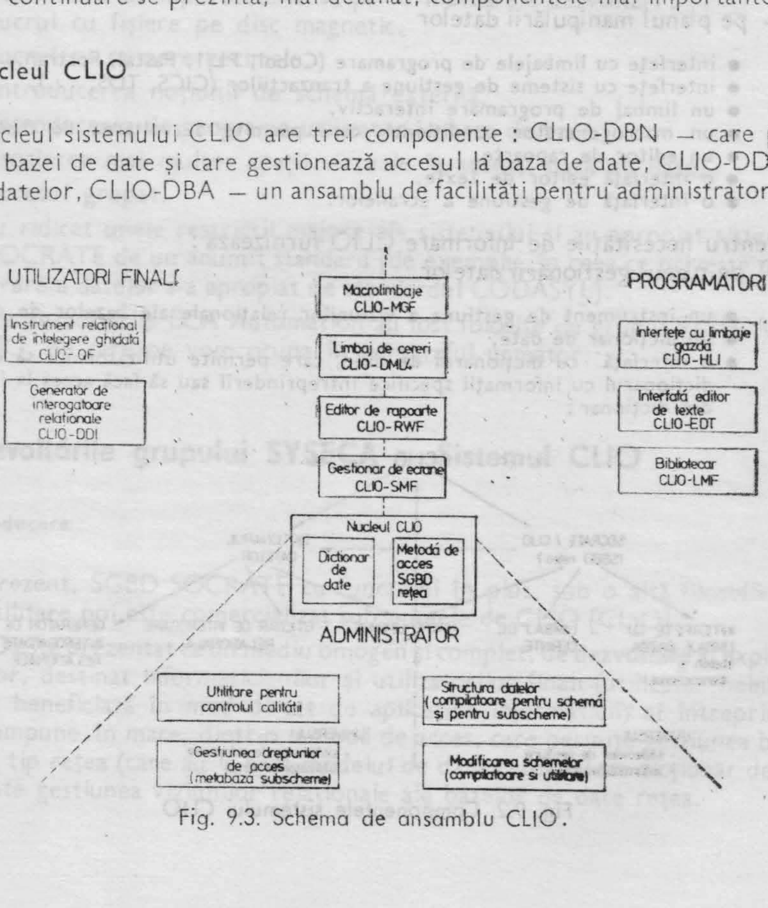


Fig. 9.3. Schema de ansamblu CLIO.

CLIO-DBN

CLIO-DBN este componenta CLIO care permite definirea bazelor de date ierarhice și rețea și gestiunea accesului la aceste baze de date (metodele de acces, controlul concurenței și jurnalizarea).

Structurarea datelor la CLIO se sprijină pe trei concepte : structurare logică, structurare fizică și spații virtuale. Grație acestei structurări, din care numai nivelul logic este văzut de programatori și utilizatori finali, CLIO permite realizarea de aplicații evolutive, independente de suportul fizic al bazei de date.

Nivelul logic. La acest nivel se definesc entitățile, atributele acestor entități, asocierile dintre entități și anumite restricții de integritate a datelor. Strategiile de acces permise sînt : acces prin cheie discriminantă sau multiplă, acces prin număr intern (data base key), acces pe asociere și acces pe inversă.

Nivelul virtual. Un spațiu virtual este un spațiu de adresare de dimensiune foarte mare, pe care este implantată, într-o manieră statică structura logică a unei baze de date. Implantarea statică este realizată rezervînd în cadrul nivelului virtual : o lungime maximă pentru fiecare atribut și numărul maxim de bytes pentru fiecare entitate.

Nivelul fizic. Acest nivel definește fișierele cu acces direct relativ, gestionate de sistemul de operare al calculatorului gazdă, pe care sînt implantate fizic datele. CLIO realizează unele optimizări la nivel fizic și anume :

- tehnici de comprimare a datelor (bit packing, codage etc.),
- structuri variabile de date (numai părțile cu informații ale spațiului virtual au o reprezentare în spațiul fizic asociat),
- alegerea implantării structurii logice pe structura fizică la nivel de atribut în una sau mai multe arii.

O bază de date conține M obiecte (nivel logic) — fiecare obiect cu atributele lui și legăturile dintre obiecte, N spații virtuale (nivelul intermediar) și P fișiere (în sensul sistemului de operare).

CLIO-DD

Dicționarul datelor CLIO-DD reprezintă o descriere centralizată a conținutului bazei de date, un instrument de documentare automată a aplicațiilor, un suport de bază al instrumentului de interogare ghidată și, împreună cu macrogeneratorul condițional, permite dezvoltarea de alte interogatoare relaționale.

Dicționarul datelor este completat automat la compilarea schemei, a structurilor externe și a cererilor de acces. Utilizatorii au acces la informațiile din dicționar și pot chiar să le completeze (identificatori externi, comentarii etc.).

Instrument de descriere. Dicționarul datelor conține identificatorii schemei, ai structurilor de date externe, ai macrocererilor și ai programelor de aplicații.

Instrument de documentare. Dicționarul datelor permite cunoașterea :

- listei tuturor programelor scrise în CLIO-DML4,
- listei identificatorilor de diferite tipuri, manipulați într-un program,
- statisticilor privind utilizarea identificatorilor datelor,
- listei programelor ce manipulează un anumit identificator,
- descrierii schemei bazei de date,
- descrierii structurilor de date externe,
- listei macroinstrucțiunilor,
- listei programelor externe scrise într-un alt limbaj decît CLIO-DML4 și apelate din programe CLIO-DML4.

CLIO-DD este o aplicaţie CLIO, numită METABASE, şi care conţine :

- o schemă de bază de date, numită metastructură. Această metastructură este invariabilă pentru toate bazele de date ;
- programe scrise în limbajul CLIO-DML4, numite metaprograme. Aceste programe permit obţinerea automată a documentaţiei ;
- date, numite metadate.

CLIO-DBA

CLIO-DBA reprezintă un ansamblu de facilităţi pentru administrarea bazelor de date, permiţînd :

— definirea şi controlul accesului la date,
— definirea şi controlul condiţiilor de utilizare a resurselor (procesoarelor) CLIO,
— definirea şi controlul integrităţii,
— adaptarea structurii logice a datelor la noile cerinţe,
— adaptarea structurii fizice a datelor (dimensiunea şi suportul fişierelor fizice, repartizarea datelor structurii logice pe fişiere fizice, algoritmii de acces) pentru a răspunde cerinţelor de performanţă,
— măsurarea costurilor de exploatare.

Controlul accesului la date se efectuează la :

- conectarea unui utilizator la o bază de date,
- compilarea cererilor,
- activarea (cerere de execuţie) cererilor precompilate şi catalogate.

Instrumentul care asigură acest control se bazează pe o metabază numită „repertoar al bazelor" şi pe conceptul de subschemă (structură externă). La nivelul entităţilor se pot defini drepturi de consultare, creare şi suprimare. La nivelul cîmpurilor se definesc drepturile de consultare şi modificare, iar la nivelul asocierilor se definesc drepturile de utilizare, creare şi suprimare. Toate aceste drepturi de acces se definesc la nivelul subschemelor. Unei subscheme i se asociază un nume şi o listă de utilizatori, care au acces la baza de date prin intermediul acelei subscheme.

Controlul utilizării resurselor. Administratorul bazei de date asociază fiecărui utilizator drepturi specifice, relativ la posibilitatea de a activa sau nu diferite procesoare CLIO, şi anume :

- de definire a schemei şi subschemelor,
- de compilare a cererilor,
- de gestiune a bibliotecilor de cereri şi macroinstrucţiuni,
- de administrare a bazei de date,
- de gestiune a metabazei.

Controlul integrităţii bazei de date. Sistemul CLIO furnizează o serie de utilitare pentru :

- controlul coerenţei căilor de acces ale CLIO şi a căilor de acces ale aplicaţiilor,
- corijarea anumitor incoerenţe ale căilor de acces ale aplicaţiilor.

Utilitare la nivelul schemei fizice. Aceste utilitare permit :

- cunoaşterea coeficientului de umplere a fişierelor fizice pe care este implantată schema logică,
- obţinerea de statistici privind diferiţi algoritmi de acces direct la date (statistici asupra funcţiilor de randomizare),
- adaptarea suporturilor şi a taliei fişierelor fizice la necesităţile de moment.

Utilitarul de schimbare a suportului salvează şi restaurează baza de date fără a perturba implantarea virtuală a structurii logice (programele, pointerii şi tabelele de acces). Tot fără a modifica implantarea virtuală a datelor pot fi realizate adăugări de atribute, asocieri şi entităţi pe spaţiul virtual existent sau pe un nou spaţiu virtual.

Celelalte componente CLIO

CLIO-DML4

CLIO-DML4 este un limbaj de manipulare a datelor, autonom, structurat şi deschis (poate fi îmbogăţit). CLIO-DML4 este un limbaj de programare complet : teste, bucle, apel de subprograme, manipulare de şiruri de caractere, gestiune de texte, editare, acces la fişiere clasice, secvenţiale, pe bandă sau disc magnetic, sortare integrată.

CLIO-HLI

Sistemul CLIO are interfaţă cu limbajele Cobol, PL/1, Fortran, Pascal şi cu limbajul de asamblare. Este o interfaţă de execuţie, apelabilă prin CALL, cu două moduri de exploatare : pe loturi şi tranzacţional.

Cererile precompilate sînt produse în mod autonom, pot fi testate independent de programele gazdă şi pot fi catalogate în cod executabil într-un spaţiu al sistemului bazei de date. Ele sînt paginate conform principiului memoriei virtuale şi sînt reentrante.

CLIO-MGF

Macrogeneratorul condiţional, CLIO-MGF, permite administratorului unei baze de date să dezvolte :

- extensii ale limbajului de cereri,
- o bibliotecă de macroinstrucţiuni, orientate aplicaţie, care să uşureze munca de programare,
- un limbaj orientat pe aplicaţie şi pe categorii de utilizatori finali.

CLIO-RWF

Editorul de rapoarte, CLIO-RWF, are două părţi :

— o parte declarativă — care corespunde definirii raportului,

— o parte executivă — care realizează lansarea şi controlul subprogramului de editare.

CLIO-SMF

Gestionarul de ecrane, CLIO-SMF, furnizează facilităţi pentru :

- definirea de machete ecran, atunci cînd nu sînt furnizate de constructor,

- interfațarea execuției tranzacțiilor, programate în CLIO-DML4, cu machetele de ecran, definite în prealabil și catalogate.

Scopul principal al acestor facilități a fost acela de a permite programarea în întregime a tranzacțiilor în limbajul CLIO-DML4.

CLIO-QF

CLIO-QF este un instrument de interogare ghidată, de tip relațional, destinat, în principal, utilizatorilor finali.

CLIO-QF este numit relațional deoarece permite :

— utilizatorului să se preocupe exclusiv de ceea ce vrea să facă și nu de cum trebuie să facă ;

— realizarea, în cadrul aceleiași interogări asupra mai multor tabele, a trei operații ale algebrei relaționale : selecția, proiecția și joncțiunea.

CLIO-QF este un instrument cu ghidare în sensul că utilizatorul :

- nu este obligat să cunoască vre-un limbaj de programare. Utilizatorul este ghidat pas cu pas, cu ajutorul mai multor machete ecran, pentru a-și formula cererea ;
- nu este obligat să cunoască schema bazei de date. El dispune de comenzi elementare, care-i permit să vizualizeze : obiectele sau tabelele, numele atributelor tabelelor și asocierile funcționale dintre tabele (informații necesare pentru operațiile de joncțiune).

Asocierea funcțională se referă la două obiecte sau tabele și un concept comun celor două tabele, tradus prin prezența aceluiași atribut în cadrul fiecărei tabele. Teoretic, prezența aceleiași coloane în două tabele este suficientă pentru a le asocia funcțional. În practică, din motive de cost de exploatare și timp de răspuns, se impune definirea în prealabil a unor drumuri de acces. Acest lucru se poate realiza în două moduri :

— pointeri fizici — tehnică utilizată la sistemele de tip ierarhic și rețea,

— pointeri simbolici — tehnică utilizată la sistemele relaționale.

CLIO-QF permite definirea în avans a asocierilor funcționale compuse (prin reuniunea asocierilor funcționale elementare), dar care sînt văzute și manipulate ca asocieri funcționale elementare. Asocierile AF1 și AF2 (vezi figura 9.4) indică două asocieri funcționale elementare (la fiecare corespunde un drum de acces). Asocierea funcțională AF3 indică o asociere funcțională între tabelele Comandă și Produs, obținută prin reuniunea asocierilor AF1 și AF2 și care va permite utilizatorului să ajungă la tabele Produs pornind de la tabele Comandă (sau invers), fără să cunoască asocierile AF1 și AF2. O parte a asocierilor funcționale sînt deduse și înregistrate automat din schema „rețea" a bazei de date. La acestea pot fi adăugate asocieri funcționale, care au la bază pointeri simbolici, și asocieri funcționale compuse.

Ghidarea pas cu pas a utilizatorului, pentru a-și formula comenzile, se face cu ajutorul mai multor machete de ecran. Fiecare machetă conține o parte de vizualizare și/sau percepere și o linie care prezintă continuările/înlănțuirile de machete ecran posibile, pornind de la ecranul curent. În fiecare moment, utilizatorul are două alternative :

- să apeleze următoarea machetă ecran, specifică ecranului curent, pentru a continua definirea cererii,
- să abandoneze cererea.

Fig. 9.4. Exemplu de asociere funcțională compusă

CLIO-QF permite utilizatorului să formuleze cereri, conform următorului scenariu :
- joncțiunea între mai multe tabele (în cadrul unei cereri pot fi joncționate pînă la 8 tabele) ;
- selecția anumitor linii din fiecare tabelă. Selecția se poate realiza pe mai multe coloane ;
- proiecția pe anumite coloane ale tabelei.

Rezultatul oricărei cereri este o tabelă.

CLIO produs portabil

CLIO este un produs portabil (a fost realizat într-un limbaj de nivel înalt). În prezent produsul este disponibil pe următoarele tipuri de calculatoare :

- BULL, modelele :
  - 64 DPS 7 sub sistem de operare GCOS64, cu interfață TDS,
  - 66 DPS 8 sub sistem de operare GCOS3, cu interfață TDS,
  - MINI 6 sub sistem de operare GCOS 400 MFS și DSS, cu interfețe DTF și DFC,
  - MITRA sub sistem de operare MMT2,
  - SOLAR sub sistem de operare TSM, RTES/D sau MPES ;
- DEC VAX subsistemele de operare VMS și UNIX
- IBM 370, 43xx, 303x, 308x, sub sistemele de operare DOS/VSE, OS/MVS și VM/CMS
- 68 000 sub sistemul de operare UNIX.

Dezvoltări distribuite

Noțiuni de baze de date distribuite

O **bază de date distribuite** este o bază de date (respectă definiția și conceptele bazelor de date), logic integrată (are o singură schemă conceptuală), dar fizic dispersată pe mai multe calculatoare, interconectate în cadrul unei rețele de calculatoare [DAVENPORT]. Noțiunile din domeniul bazelor de date rămîn valabile pentru bazele de date distribuite, iar pentru acestea din urmă, apar o serie de noțiuni noi și anume : fragmentarea, replicarea, controlul distribuit, blocarea și realizarea în două faze, mărci de timp și inel virtual.

Fragmentarea (partiționarea). Baza de date (o mulțime de colecții de date și legăturile dintre ele) este fragmentată conform principiilor :
- plasarea datelor memorate în mediul de producere și utilizare a lor,
- minimizarea transportului de date prin rețeaua de calculatoare.

Fragmentarea, pentru a răspunde acestor principii, se realizează la două nivele :
- partiționarea mulțimii de colecții de date în submulțimi de colecții de date,
- partiționarea unei colecții de date în fragmente.

Fragmentarea colecției de date se poate realiza în două moduri :
- orizontal — fragmentele rezultate au aceeași structură ca și colecția, dar diferă între ele prin datele pe care le conțin,
- vertical — fragmentele rezultate conțin doar o parte, fiecare, din structura colecției.

Uneori se admite şi combinarea celor două moduri. Fragmentele rezultate constituie elementele de distribuire a datelor. Totalitatea fragmentelor unei baze de date distribuite, memorate pe un nod al reţelei de calculatoare formează o bază de date locală.

Replicarea. Am afirmat că bazele de date distribuite respectă toate principiile bazelor de date, inclusiv pe cel al redundanţei. Totuşi, pentru a mări disponibilitatea accesului la date, bazele de date distribuite admit un anumit grad de redundanţă (replicarea datelor sau copii multiple ale fragmentelor de date) controlată. Menţinerea unei copii de fragment se justifică (din punct de vedere economic) numai dacă se menţine inecuaţia :

$$C_{mc} + \sum_{i=1}^{p} C_a l_i + C_a < \sum_{i=1}^{p} C_a d_i$$

unde :

C_{mc} este costul de memorare a copiei ;
$C_a l_i$ – costul accesului local pentru utilizatorul i ;
C_a – costul de actualizare a copiei ;
$C_a d_i$ – costul accesului la distanţă (la copia primară) pentru utilizatorul i.

Gestiunea copiilor multiple este realizată de către sistemul de gestiune a bazei de date distribuite (SGBDD). Actualizarea copiilor multiple se poate face în două moduri:

- simultan,
- actualizarea copiei primare şi aducerea la zi a copiilor secundare cu o anumită întîrziere.

Controlul distribuit. Cererile de acces pot solicita date situate pe noduri diferite ale reţelei de calculatoare. Pentru a superviza execuţia unor astfel de cereri, un nod trebuie să-şi asume rolul de coordonator. Celelalte noduri, care participă la execuţia acelei cereri, sînt noduri cooperante. Un nod poate să fie în acelaşi timp coordonator (pentru cererile lansate din acel nod) şi cooperant (pentru cereri lansate de la alte noduri dar care solicită acces la acest nod). Controlul execuţiei cererii la distanţă se realizează prin schimb de mesaje între coordonatorul cererii şi cooperanţii acelei cereri.

Blocarea şi realizarea în două faze. Pentru a se evita introducerea de inconsistenţe în baza de date distribuită, în condiţii de concurenţă, se recurge la blocarea datelor (noţiune folosită şi la bazele de date centralizate). Elementul de noutate la bazele de date distribuite îl reprezintă blocarea în două faze, care are la bază următoarele principii :

– nici-o cerere nu are acces la date dacă acestea nu au fost blocate în prealabil (blocate în citire sau în actualizare) ;
– după ce a deblocat un element de dată, acea cerere nu mai poate bloca nici-un element de dată.

Pentru a nu introduce inconsistenţe în baza de date distribuită, de data aceasta din cauza diferitelor incidente hardware sau software, se utilizează tehnica de realizare în două faze. Acest tip de realizare împarte execuţia unei cereri de actualizare în două faze :

- pregătirea execuţiei – citirea datelor, scrierea în jurnalele locale a „imaginilor înainte" (date neactualizate) şi a „imaginilor după" (datele actualizate) ;
- realizarea sau nerealizarea cererii – introducerea actualizărilor în baza de date distribuite, cînd sistemul funcţionează corect, sau înlăturarea efectelor acestei cereri de actualizare cînd apare un incident.

Nodul coordonator, pentru acea cerere, trimite mesajul „PREGĂTEŞTE" tuturor nodurilor cooperante. Fiecare din nodurile cooperante realizează faza unu la primirea mesajului şi returnează mesajul „PREGĂTIT" la coordonator. Cînd primeşte acest mesaj de la toate nodurile cooperante, nodul coordonator le transmite mesajul „REALIZEZA". La primirea acestui mesaj nodul cooperant realizează faza a doua şi transmite coordonatorului mesajul „REALIZAT". La primirea tuturor acestor mesaje, coordonatorul consideră cererea executată cu succes. Dacă pe perioada acestui proces apare un incident, atunci se emite un mesaj „ABORT" şi toate nodurile cooperante acţionează pentru a se reveni la situaţia de dinaintea lansării acelei cereri. Acest mecanism garantează că fie cererea este executată corect peste tot, fie că nu se execută deloc nicăieri.

Mărci de timp şi inel virtual. O cerere de acces la baza de date distribuită poate solicita acces la mai multe baze de date locale. În condiţii de concurenţă trebuie ca cererile să se execute, în ordinea emiterii lor, la toate nodurile implicate. Pentru a realiza acest lucru au fost inventate mai multe mecanisme, dintre care cele mai cunoscute sînt mărcile de timp (timestamps) şi inelul virtual.

Mărcile de timp. Fiecare cerere, la emiterea ei, primeşte automat o marcă de timp (identificatorul nodului plus timpul ceasului local). Fiecare articol din baza de date distribuită are o marcă de timp care se aduce la zi la fiecare actualizare cu marca de timp a cererii care a actualizat-o. Sistemul de gestiune asigură execuţia cererilor în ordinea crescătoare a mărcilor de timp ; o cerere este executată dacă marca sa de timp este mai mare decît marca de timp a articolelor de date la care solicită accesul. Pentru a se asigura un ceas unic pe întreaga reţea, la recepţionarea unei cereri nodul respectiv verifică ora locală cu ora din marca de timp a cererii primite. Dacă ora locală este mai mică atunci se actualizează ora locală. În acest fel în scurt timp se ajunge practic la o oră unică pe reţea.

Inel virtual. Nodurile reţelei sînt înlănţuite logic, formînd un inel virtual, în care fiecare nod are un predecesor şi un succesor bine stabilit. Pe acest inel circulă continuu un tichet. Nodul la care se află tichetul, dacă acesta este liber, poate să emită un mesaj. Tichetul apoi trece pe la toate nodurile şi mesajul este preluat de nodurile cooperante. Cînd tichetul ajunge din nou la nodul care a emis mesajul, tichetul devine liber şi pleacă la nodul următor.

Pentru o documentare mai amplă în domeniul bazelor de date distribuite recomandăm următoarea bibliografie [BERKELEY, BERLIN, COJOCARU83, COJOCARU84, COJOCARU85, GARDARIN82, INFOTECH, LINDSAY, MOHAN, STONEBRAKER].

Variante de distribuire pentru SGBD SOCRATE

Realizarea unor versiuni distribuite pentru SGBD SOCRATE a fost luată în considerare de timpuriu [CHUPIN]. Au fost experimentate mai multe soluţii, care acoperă o gamă largă de abordări : SOCRATE distribuit, distribuirea unui subset al limbajului şi modelului de date SOCRATE, (sistemul FRERES [BOSC]), SOCRATE ca sistem subordonat (sistemul PLEXUS [ZURLUH]), SOCRATE ca sistem de gestiune locală în cadrul unui sistem distribuit (sistemul POLYPHEME [ADIBA79, ADIBA80, DECITREa, DECITREb, DELOBEL]), facilităţi de acces la distanţă din programe SOCRATE (SOCRATE-mini [MIDAS]). În general, prototipurile realizate

(cu excepţia lui POLYPHEME) nu au reuşit să se impună şi acest lucru, în special, din cauza dificultăţilor modelului de date SOCRATE vis-a-vis de distribuire, în raport cu modelul relaţional al datelor.

PLEXUS este un prototip de sistem de gestiune a bazelor de date distribuite, elaborat la Universitatea Paul-Sabatier din Toulouse. Sistemul asigură gestiunea unei baze de date distribuite pe o reţea de microcalculatoare SOLAR 16. Accesul la datele locale se face prin intermediul lui SOCRATE VI.6. În cadrul sistemului PLEXUS, datele sînt structurate pe următoarele niveluri:

- scheme externe,
- schema conceptuală globală,
- schema internă globală,
- scheme conceptuale locale.

PLEXUS prezintă o arhitectură funcţională multistrat. La tratarea unei cereri intervin trei module:

— modulul tranzacţional (MT),
— modulul de control (MC),
— modulul de execuţie (ME).

Fiecărui modul îi este atribuită o funcţie la nivel global şi o funcţie la nivel local (figura 9.5). Produsul PLEXUS implementează mare parte din conceptele prezentate anterior (controlul distribuit, blocarea şi realizarea în două faze, inelul virtual şi replicarea).

POLYPHEME este un prototip de sistem de gestiune a bazelor de date distribuite eterogene. Sistemul a fost implementat pe reţele de calculatoare CYCLADES şi TRANSPAC. Ceea ce este caracteristic sistemului POLYPHEME, în raport cu alte SGBD-uri, este faptul că el asigură cooperarea unor baze de date existente, gestionate cu sisteme locale eterogene (SOCRATE, IMS, CODASYL, ...). Această cooperare se exprimă prin posibilitatea definirii unei scheme (globale) unice care să descrie ansamblul bazelor de date, integrate în acest fel într-o bază de date distribuite.

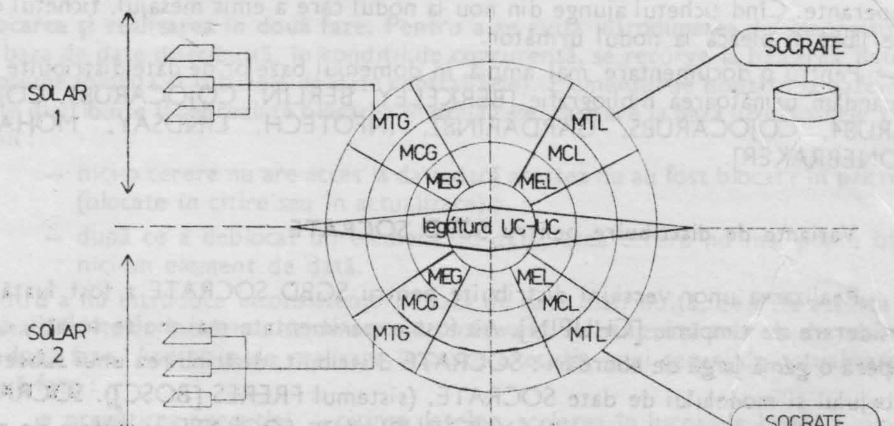


Fig. 9.5. Arhitectura funcţională a sistemului PLEXUS

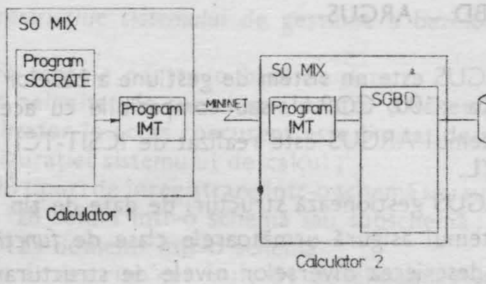


Fig. 9.6. SOCRATE-mini în regim de teleprelucrare utilizînd o reţea de calculatoare

Una din problemele esenţiale ale sistemului POLYPHEME este traducerea cererilor pentru scheme globală în cereri pentru schemele locale. La nivel global a fost utilizat modelul relaţional, utilizîndu-se sistemul URANUS. URANUS este un sistem de baze de date relaţionale, care oferă următoarele facilităţi :

— un limbaj relaţional de definire a datelor,
— un limbaj de manipulare a datelor bazat pe algebra relaţională CODD,
— posibilităţi de stocare şi acces la date relaţionale, date ce pot fi extrase dintr-o bază de date nerelaţională (de exemplu, SOCRATE).

Uranus poate funcţiona şi ca SGBD autonom şi ca interfaţă relaţională pentru SGBD SOCRATE.

SOCRATE mini (vezi volumul 2) este un SGBD realizat la CTCE Constanţa pornind de la facilităţile SOCRATE V1.5. La ultima sa versiune V3, SOCRATE-mini oferă o facilitate de acces la distanţă din programe de aplicaţii SOCRATE. Dintr-un program SOCRATE se poate apela un program, care, prin perechea sa de la distanţă, poate avea acces la o altă bază de date (figura 9.6). SGBD-ul de la distanţă poate să fie de orice tip. Cele două programe, inclusiv lucrul cu reţeaua, cad în sarcina utilizatorului.

Dezvoltarea tuturor acestor versiuni a avut la bază următoarele motivaţii :
— ridicarea restricţiilor şi anomaliilor versiunilor mai vechi,
— adăugarea de noi facilităţi,
— alinierea la standardele internaţionale,
— asigurarea independenţei datelor (structurarea datelor pe mai multe nivele),
— realizarea unui sistem pentru o categorie cît mai largă de utilizatori (adoptarea modelului relaţional).

Realizări de SGBD pe minicalculatoare în ţara noastră

Ca urmare a extinderii utilizării minicalculatoarelor în economie a crescut interesul pentru dezvoltarea componentelor software pe minicalculatoarele de producţie românească din familia I-100 şi CORAL.

În acest context au fost realizate, testate, implementate şi comercializate cel puţin următoarele SGBD pe minicalculatoare : SOCRATE-MINI (MIDAS), ARGUS şi LEDA.

Întrucît SGBD SOCRATE-MINI constituie obiectul volumului 2 al acestei lucrări, considerăm binevenită o prezentare sumară şi a celorlalte două sisteme ARGUS şi LEDA. La acest mod, cititorul va putea să-şi formeze o imagine mai completă asupra SGBD pe minicalculatoare realizate în ţara noastră.

SGBD – ARGUS

ARGUS este un sistem de gestiune a bazelor de date pentru minicalculatoare din familia I-100, CORAL sau compatibile cu acestea.

Sistemul ARGUS este realizat de ICSIT-TCI, în conformitate cu specificaţiile CODASYL.

ARGUS gestionează structuri de date de tip ierarhic şi reţea.

Sistemul asigură următoarele clase de *funcţiuni* :

— descrierea diverselor nivele de structurare a datelor ;
— accesul la datele şi relaţiile bazei de date ;
— protecţia datelor ;
— refacerea stării bazei de date în caz de incident ;
— auxiliare.

Componentele sistemului ARGUS sînt :

SCH. TSK – Taskul de compilare a schemei sursă a bazei de date ;
DIC. TSK – Taskul de creare a dicţionarului datelor ;
SBS. TSK – Taskul de compilare a unei subscheme sursă şi crearea subschemei obiect ;
DSH. TSK – Taskul de descriere interactivă a schemei bazei de date şi compilarea schemei ;
DSS. TSK – Taskul de descriere interactivă a subschemelor şi crearea de subscheme obiect ;
VSD. TSK – Taskul de proiecţie (Mapare) a spaţiului virtual al BD în spaţiul fizic ;
INT. TSK – Taskul de iniţializare a bazei de date şi extindere a domeniilor bazei de date ;
ARG. TSK – Taskul de tratare a comenzilor LMD ;
SGL. TSK – Taskul de realizare a operaţiilor de intrare/ieşire ;
LDT. TSK – Taskul satelit al SGL-ului ;
LID. TSK – Taskul de listare şi modificare a conţinutului dicţionarului datelor ;
LSB. TSK – Taskul de listare şi modificare a conţinutului unei subscheme obiect ;
STP. TSK – Taskul de terminare forţată a execuţiei taskurilor ARG şi SGL ;
ARG. CMD – Fişier de comenzi indirecte pentru instalarea taskurilor ARGUS ;
LMD. TSK – Taskul care execută precompilarea programelor de acces la baza de date scrise în COBOL + LMD ;
LMF. TSK – Taskul care acceptă precompilarea programelor de acces la baza de date descrise în FORTRAN + LMD ;
RBJ. TSK – Taskul care execută refacerea bazei de date după un incident ;
BDS. TSK – Taskul care execută editarea de situaţii statistice privind domeniile bazei de date.

În prezent realizatorul ARGUS elaborează versiunea 3 care cuprinde următoarele *facilităţi suplimentare* :

— utilitar de verificare şi refacere a consistenţei bazei de date ;
— utilitar pentru reorganizarea bazei de date ;
— modul pentru gestiunea unei baze de date distribuită pe o reţea omogenă de minicalculatoare ;
— utilitar pentru descrierea interactivă a subschemelor folosite de interfaţa ARGUS-DATARIEVE (taskul DMS).

Performanțele, limitele și restricțiile sistemului de gestiune a bazelor de date ARGUS sînt:

— descrierea unei singure scheme pentru o bază de date;
— descrierea unui număr nelimitat de subscheme pentru o schemă;
— numărul taskurilor utilizator în acces concurent este limitat de restricțiile sistemului de operare și a configurației sistemului de calcul;
— descrierea a cel mult 128 tipuri de înregistrare într-o schemă sau subschemă;
— descrierea a cel mult 128 seturi într-o schemă sau subschemă;
— descrierea a cel mult 125 domenii într-o schemă;
— într-o înregistrare pot fi descrise cel mult 64 de cîmpuri (elementare, grup, vector);
— cheia unei înregistrări poate fi formată din cel mult 5 cîmpuri (elementare sau grup);
— lungimea cheii înregistrărilor poate avea cel mult 64 de caractere;
— numărul de realizări a unui tip de înregistrare nu poate depăși 2 000 000;
— lungimea unei realizări de tip înregistrare nu poate fi mai mare de 1 024 bytes;
— lungimea unui cîmp nu poate fi mai mare de 512 bytes;
— lungimea unei condiții de validare nu poate fi mai mare de 512 bytes;
— într-un set nu pot exista mai mult de 5 tipuri de înregistrare declarate MEMBER;
— sortarea membrilor într-o realizare de set nu se poate face pe mai mult de o cheie de sortare;
— numărul suporturilor disc ale bazei de date nu este limitat;
— numărul de pagini a unui domeniu nu poate fi mai mare de 65 000;
— dimensiunea unei pagini nu poate fi mai mare de 8k(b);
— asigură interfața cu programe scrise în limbajele COBOL și FORTRAN;
— asigură interfața ARGUS-DATATRIEVE.

Timpul de acces este redat în tabelele 9.2, 9.3, iar resursele hardware sînt redate în tabelul 9.1.

Sistemul de gestiune a bazelor de date ARGUS costă 65 000 lei (neincluzînd costul cursurilor de pregătire).

Odată cu livrarea produsului, realizatorul asigură asistență tehnică pentru:
— instalarea produsului;
— realizarea de aplicații care utilizează sistemul ARGUS.

Documentația livrată odată cu produsul cuprinde:
— Manualul de prezentare — ARGUS;
— Manualul de utilizare — Limbaje de descriere a datelor;
— Manualul de utilizare — Limbaje de manipulare a datelor;
— Manual de utilizare — INTERFAȚA ARGUS-DATATRIEVE;
— Manual — Ghidul de instalare și operare;
— Exemplu de control pentru ARGUS.

SGBD — LEDA

LEDA — Large Endowment for Data Administration — este un sistem interactiv de tip CODASYL, de gestiune a bazelor de date pentru minicalculatoare din familia I-100, CORAL sau compatibile cu acestea. Este realizat în cadrul Institutului de Tehnică de Calcul și Informatică (ICSIT-TCI).

SGBD LEDA asigură următoarele *funcţiuni* :

— gestiunea bazelor de date cu structuri de date arborescente şi reţea ;

— descrierea datelor cu ajutorul limbajelor de descriere a schemei şi subschemei ;

— realizarea de aplicaţii utilizînd limbajul de manipulare a datelor ;

— accesul conversaţional la baza de date prin intermediul limbajului de interogare LDA. În cadrul acestuia pot fi declarate variabile ce pot fi utilizate în diferite calcule ;

— vizualizarea schemei şi subschemelor bazei de date (utilitarul EFL) ;

— iniţializarea bazei de date (utilitarul INL) ;

— obţinerea de statistici privind conţinutul unor arii sau pagini ale bazei de date (utilitarul EDL) ;

— restaurarea bazei de date.

Sistemul LEDA are următoarele *componente* :

1. LEDA-DDL — Limbajele administratorului bazei de date.

SDL — Limbajul de descriere a schemelor (Schema Description Language) ;
SDDL — Limbajul de descriere a subschemelor (Subschema Data Description Language);
DMCL — Limbajul de descriere a interfeţei fizice (Device-Media Control Language).

2. LEDA-UTL — Utilitarele bazei de date

RFL — Iniţializarea repertoriului ;
EFL — Editarea fişierului repertoriu ;
INL — Iniţializarea fişierelor bazei de date ;
SVL/ESL — Salvarea/restaurarea fişierelor bazei de date ;
JLS — Listarea jurnalului bazei de date ;
JCM — Comprimarea mai multor fişiere jurnal într-un singur fişier ;
JRS — Restaurarea bazei de date pornind de la fişierul jurnal) ;
DBR — Restaurarea bazei de date ;
RGN — Generator de rapoarte ;
EDL — Editor pentru fişiere tip LDA ;
VRF — Verificarea înlănţuirilor din baza de date specificată ;
APL — Set de fişiere de comenzi indirecte ;
QRK — Recuperarea imediată din avarie.

3. LEDA-APL — Limbajele programării aplicative

DML — Limbajul de manipulare a datelor pentru limbajele gazdă COBOL, FORTRAN, C şi MACRO ;
LDA — Limbaj conversaţional de manipulare a datelor.

Enumerarea acestor componente permite formarea unei imagini mai ample asupra posibilităţilor pe care le oferă sistemul LEDA.

Performanţele, limitele şi restricţiile sistemului LEDA sînt :

— organizarea bazei de date pe suporturi multivolum ;

— accesul concurent a cel mult 30 de utilizatori pe aceeaşi bază de date ;

— permite 300 de fişiere pentru înregistrările bazei de date (numărul poate fi extins) ;

— pînă la 16 milioane articole într-un fişier al bazei de date ;

— număr nelimitat de cîmpuri pe articol ;
— lungimea maximă a unui cîmp este de 250 octeți ;
— o subschemă nu admite numai descrierea tipului de articol și cere obligatoriu și o legătură de SET ;
— în descrierea setului în SCHEMA bazei de date LEDA nu admite SET pe același tip de articol conform specificațiilor CODASYL (adică același tip de articol să fie și OWNER și MEMBER) ;
— în descrierea a două seturi pe aceleași tipuri de articole OWNER și MEMBER LEDA nu admite două seturi declarate AUTOMATIC ci unul trebuie să fie declarat manual conform specificațiilor CODASYL ;
— modul de memorare a articolelor în arii se face după o cheie de randomizare (modul calcul) ceea ce duce la dispersarea articolelor în arie fără o ordonare a cheii iar regăsirea se face în ordinea așezării fără posibilitatea de sortare după cheia CALC. Acest lucru duce la imposibilitatea listării articolelor în ordinea unei anumite chei. Cînd modul de memorare este direct articolele se stochează în ordinea citită din fișierul de intrare, dar regăsirea articolelor după o cheie de selecție se poate face numai secvențial ;
— confidențialitatea înregistrărilor și jurnalizarea articolelor ;
— acces simultan și facilități de actualizare în condițiile multiutilizator.

În ceea ce privește performanțele sub aspectul timpului de acces precum și resursele hardware vor fi redate în paragraful, Resurse și performanțe ARGUS-LEDA, SOCRATE-MINI.

Sistemul de gestiune a bazelor de date LEDA costă 89 000 lei, preț în care intră asistența tehnică oferită de producător pe timp de 1 an și documentația, care cuprinde :
— Ghid LEDA ;
— Memento LEDA ;
— Manualul de programare — Limbajul de descriere a datelor și a organizării lor logice ;
— Manualul de programare — Limbajul de manipulare a datelor ;
— Manualul de prezentare și utilizare — Programe utilitare ;
— Ghid de instalare și lansare.

Sistemul este livrat pe bandă magnetică și conține :
— Taskurile sistemului LEDA ;
— Modulele obiect ale sistemului LEDA ;
— Fișierele de comenzi indirecte.

Resurse și performanțe ARGUS, LEDA, SOCRATE-MINI*

În cele ce urmează vom reda doar cîteva elemente de comparație pentru cele trei sisteme, astfel :
— sînt destinate gestiunii bazelor de date pe minicalculatoare ;
— ARGUS și LEDA sînt elaborate în concordanță cu specificațiile CODASYL pentru baze de date ;
— SOCRATE-MINI este elaborat în concordanță cu specificațiile SOCRATE ale C.I.I. ;
— ARGUS și LEDA acceptă interfețe cu limbajele COBOL și FORTRAN ;

*) Menționăm că rezultatele au caracter orientativ, testările fiind făcute de colective diferite, în condiții deosebite (relativ apropiate).

– SOCRATE-MINI acceptă interfeţe cu toate limbajele care respectă convenţia de apel RSX ;

– resursele hardware minime necesare implementării sînt redate în tabelul 9.4.

Tabelul **9.4. Resurse hardware minime necesare implementării**

Denumirea resursei	Tip SGBD		
	ARGUS	LEDA	SOCRATE-MINI
– Memorie internă	64 K0	64 K0	120 K0
– Unităţi de discuri	1	1	1
– Unităţi de bandă	1	1	1
– Terminale	1	1	1
– Unităţi de imprimantă	1	1	1

– Timpul de acces utilizator pe operaţii asupra datelor este redat în tabelul 9.5.

Tabelul **9.5. Timp de acces utilizator pe operaţii asupra datelor***

Operaţie	Mod de acces	Precizări	Timp necesar		
			ARGUS	LEDA	SOCRATE-MINI
1	2	3	4	5	6
ÎNCĂRCARE ADĂUGARE		*Obs*. Timpul de memorare a unei înregistrări depinde de : nr. de seturi la care participă înregistrarea, nr. de cîmpuri din structura înregistrării, regimul de lucru (batch sau conversaţional) dacă se lucrează cu jurnal activ sau nu etc. a) Încărcarea unei înregistrări în regim conversaţional în funcţie de numărul de cîmpuri	1''–2''	1''–2''	1''–2''
		b) Încărcarea a 200 de înregistrări (ARGUS şi SOCRATE) şi 100 (LEDA) în regim batch, fără a aparţine la un SET (ANNEAU)	45''	90''	11''
		c) Introducerea a 38 de înregistrări cu structuri arborescente, participante la 3 legături de SETURI AUTOMATICE în regim „multi-user" şi fără jurnal activ (pentru SOCRATE ataşarea la set s-a făcut explicit)	45''20	5'30''18	6''

* Pentru ARGUS şi LEDA datele ne-au fost puse la dispoziţie de către Centrul de calcul electronic al M.I.E.T. Bucureşti, obţinute pe baza testelor efectuate de un colectiv format din : M. Anton, L. Botezatu, E. Ziptzer, GH. Caraman, L. Leu, L. Maxim, C. Cosma, G. Sarbu, O. Stănciulescu.

Pentru SOCRATE-MINI datele ne-au fost puse la dispoziţie de către Centrul Teritorial de Calcul Electronic Constanţa.

Tabelul 9.5 (continuare)

1	2	3	4	5	6
		d) Introducerea în baza de date a 218 înregistrări membru la o realizare de SET AUTOMATIC şi cu structură arborescentă, în regim „multi-user" (pentru SOCRATE ataşarea s-a făcut explicit)	2'43''70	1 h 9'	29''
		e) Încărcarea a 41 înregistrări de lungime 17 octeţi în regim „multi-user" fără participare la SET (ANNEAU) şi fără jurnal activ	5''09	1'09''19	2''
MODIFICARE	CHEIE	Modificarea tuturor cîmpurilor cu excepţia cheii	2''	2''	0,041''
		Timpul de acces la înregistrarea solicitată	1''	1''	0,037''
STERGERE	CHEIE	Ştergerea unei înregistrări din baza de date. Ştergînd o înregistrare de tip OWNER se şterg implicit toate înregistrările de tip MEMBER din cadrul SETULUI respectiv N specifică numărul de înregistrări de tip MEMBER	max. 2''	max. 2''	0,3''*N
INTEROGARE	SECVENTIAL	Timpul de regăsire şi listare într-un fişier de tip LST a 100 înregistrării	1''		12''
	CHEIE	Regăsirea unei înregistrări	1''	1''	0,037''
	CRITERII MULTIPLE	a) Regăsirea unei înregistrări de tip OWNER cînd înregistrarea curentă este de tip MEMBER	1''	1''	0,035''
		b) Regăsirea membrilor unei realizări de tip SET cînd înregistrarea curentă este de tip OWNER N specifică numărul de înregistrări de tip MEMBER	1''*N	1''*N	0,034''*N
		c) Regăsirea a două înregistrări de tip OWNER, participante la două SETURI diferite pentru care înregistrarea curentă era de tip MEMBER pentru ambele seturi. Deci regăsirea şi listarea acestor 3 înregistrări	2''	2''	—

— Durata de trecere a unei lucrări prin sistem pe etape este redată în tabelul 9.6.

Tabelul 9.6. **Durata de trecere a unei lucrări prin sisteme de etape***

Nr. crt.	Etapa	Timp necesar		
		ARGUS	LEDA	SO-CRATE-MINI
1	Compilarea schemei şi crearea Dicţionarului datelor — cu listarea schemei — fără listarea schemei	 1' 45''	 1' 45''	 34'' 21''
2	Compilarea subschemelor şi crearea subschemelor obiect (timp pe fiecare subschemă)	20''	20''	
3	Proiecţia spaţiului virtual în spaţiul fizic (în dialog cu utilizatorul)	2'		
4	Compilarea interfeţei fizice (DEVICE-MEDIA) se realizează cu taskul DDL care are drept efect introducerea lui DEVICE-MEDIA în fişierul repertoriu Obţinerea modulului obiect al interfeţei fizice se realizează cu taskul DMG, iar crearea taskului DEVICE-MEDIA se realizează cu DVM — compilarea cu DDL — compilarea cu DMG — crearea taskului cu DVM		 20'' 30'' 1'	
5	Iniţializarea bazei de date, presupune iniţializarea cu zerouri a fişierelor bazei de date — pentru LEDA un fişier de 1 000 blocuri — pentru ARGUS şi SOCRATE un fişier de 3 000 blocuri	3'	2'	1'30''
6	Precompilarea unui program de aplicaţie indiferent de mărimea sau complexitatea acestuia (ARGUS, LEDA) Pentru SOCRATE-MINI funcţie de complexitatea şi mărimea programului	max. 1'	max. 2'	1''—30'
7	Compilarea unui program de aplicaţie cu ajutorul compilatorului limbajului gazdă (în condiţiile în care sistemul nu este prea încărcat şi programul nu este prea mare)	2'	3'	
8	LINK — editarea unui program de aplicaţie	3'—5'	3—'7'	

* Datele se bazează pe aceeaşi sursă ca şi cele din tabelul 9.5.

Din prezentarea performanţelor, limitelor şi restricţiilor celor trei sisteme se mai pot desprinde o multitudine de asemănări şi deosebiri.

Aplicație complexă cu SGBD-SOCRATE

10. PROIECTAREA BAZEI DE DATE PENTRU ACTIVITATEA DE PERSONAL

Metodica de proiectare a unei baze de date

Proiectarea unei baze de date presupune o succesiune logică de activități ce pot fi grupate în etape. Pe parcursul realizării bazei de date se întocmește o documentație avînd ca scop, pe de o parte, facilitarea comunicării între colectivele ce concură la realizarea bazei de date, iar pe de altă parte, de a permite implementarea, exploatarea și întreținerea bazei de date. Documentația reflectă soluțiile date problemelor implicate și ea este supusă spre avizare la încheierea anumitor etape.

La realizarea activității de proiectare a bazei de date participă informaticieni (analiști, ingineri de sistem, programatori și operatori), conducerea unității economice în calitate de decident și beneficiar al sistemului informatic și personal-muncitor din compartimentele beneficiare. Fiecare dintre aceste categorii de personal avînd sarcini bine determinate.

În cele ce urmează sintetizăm sub formă tabelară activitățile ocazionate de realizarea și implementarea unei baze de date. În tabel se evidențiază etapele, activitățile din cadrul fiecărei etape precum și responsabilul fiecărei sarcini.

Procedura de realizare și implementare a unei baze de date

Pas activitate	Descrierea activității	Cine răspunde
0	1	2
1	*Definirea scopului proiectului bazei de date*	
	a. Definirea obiectivului urmărit prin implementarea bazei de date	CCRBD
	b. Determinarea sferei de cuprindere a activităților ce vor fi servite de baza de date	CMRBD
	c. Identificarea subdiviziunilor organizatorice și definirea funcțiilor care în cadrul unității vor utiliza baza de date (identificarea relațiilor interne și externe ale unității economice referitoare la activitatea ce va utiliza baza de date)	CMRBD
	d. Identificarea aplicațiilor existente și planificate care vor fi convertite în sistemul bazei de date	CMRBD
	e. Pregătirea propunerilor pentru conducerea unității și obținerea acordului de continuare	CMRBD
	f. Avizarea propunerilor	CTE

(continuare)

0	1	2
2	*Organizarea proiectului bazei de date*	
	a. Alegerea (numirea) utilizatorilor pentru echipa de proiectare	CCRBD
	b. Selectarea administratorului bazei de date (ABD)	CCRBD
	c. Elaborarea planului iniţial şi graficului de implementare a bazei de date	ABD
3	*Alegerea sistemului de gestiune a bazei de date*	
	a Studierea cerinţelor informaţionale si identificarea tipurilor de structuri de date implicate	ABD
	b. Stabilirea criteriilor de alegere a unui SGBD	ABD
	c. Inventarierea SGBD existente	ABD
	d Alegerea propriu-zisă a SGBD din cadrul celor canditate	ABD
4	*Proiectarea structurii conceptuale a bazei de date*	
	a. Identificarea şi analiza detaliată a informaţiilor necesare conducerii (ieşirile sistemului)	AS
	b. Identificarea şi analiza datelor de intrare (intrările sistemului) necesare satisfacerii situaţiilor de informare-raportare necesare conducerii	AS
	c. Determinarea colecţiilor (entităţilor) de date şi a relaţiilor dintre ele	ABD
	d. Specificarea strategiilor de căutare în funcţie de nevoile diverselor aplicaţii şi de specificul acestor aplicaţii	ABD
	e. Specificarea modelelor de acces şi cheilor de acces pentru a asigura confidenţialitatea şi integritatea datelor	ABD
	f. Elaborarea schemei de ansamblu a proiectului bazei de date	ABD
	g. Întocmirea specificaţiilor de proiect	AS
	h. Avizarea resurselor necesare realizării bazei de date	CTE
5	*Estimarea resurselor necesare realizării bazei de date*	
	a. Identificarea fişierelor ce vor fi convertite	CAS
	b. Identificarea programelor specifice aplicaţiilor	CAS
	c. Estimarea necesarului de programatori	CAS
	d. Estimarea necesarului de purtători tehnici de informaţi i	CAS
	e. Estimarea configuraţiei de sistem	ABD şi INGS
	f. Dezvoltarea graficului de realizare şi implementare	ABD
6	*Realizarea propriu-zisă a bazei de date în conformitate cu specificaţiile SOCRATE*	
	a. Generarea bazei de baze	ABD
	b. Generarea bazei de date şi formatarea acesteia	ABD
	c. Definirea structurii conceptuale a bazei de date utilizînd LDD	ABD
	d. Elaborarea programelor de încărcare, exploatare şi actualizare a bazei de date, utilizînd LMD	CP
	e. Completarea specificaţiilor de proiect	CAS şi CP

(continuare)

0	1	2
7	*Testarea și experimentarea bazei de date*	
	a. Testarea lanțurilor de programe și depanarea acestora	CP
	b. Încărcarea propriu-zisă a bazei de date	ABD
	c. Conversia aplicațiilor existente	CAS și CP
	d. Testarea aplicațiilor noi	CAS și CP
	e. Estimarea performanțelor bazei de date	ABD
	f. Actualizarea și definitivarea documentației de proiect	CAS și CP
	g. Avizarea proiectului	CTE
8	*Exploatarea curentă și întreținerea bazei de date*	
	a. Exploatarea curentă a bazei de date	B
	b. Ținerea la zi a bazei de date	ABD
	c. Urmărirea performanțelor bazei de date	ABD
	d. Reorganizarea bazei de date	ABD

Notații:

CCRBD — Comisia de coordonare a realizării bazei de date. Este constituită de către organul de conducere colectivă a unității beneficiare și este compusă din: șefi de compartimente, factori de răspundere din compartimentele de bază, reprezentanți ai unității de proiectare (director, director tehnic, șefi de atelier, șeful de proiect (administratorul bazei de date)).
CMRBD — Colectivul mixt de realizare a bazei de date. Este format din analiști, programatori, specialiști din compartimentele implicate în realizarea și exploatarea bazei de date.
CTE — Comisia tehnico-economică de avizare.
ABD — Administratorul bazei de date (responsabilul de proiect).
CAS — Colectivul de analiști de sistem.
CP — Colectivul de programatori.
INGS — Ingineri de sistem.

Din cele prezentate se poate observa că procedura de realizare și implementare a unei baze de date nu este identică cu metodologia generală de realizare a sistemelor informatice. Ea se regăsește în cadrul general al acesteia însă are un caracter mai restrîns, și anume, se referă în mod deosebit la soluția de organizare a datelor. Desigur o parte din activități coincid.

Pentru aplicația noastră, din considerente de spațiu, nu vom insista asupra tuturor activităților enunțate, însă, în mare parte, se va urmări procedura generală de realizare și implementare a unei baze de date.

Obiectivul activității de personal

Activitatea de personal din cadrul unităților economice are drept scop *utilizarea rațională a forței de muncă* în concordanță cu obiectivele planului de dezvoltare economico-socială, ținîndu-se seama de pregătirea și capacitatea profesională a fiecărei persoane care se încadrează în muncă și de cerințele unitare stabilite pentru fiecare loc de muncă.

Determinarea sferei de cuprindere a activității de personal

Activitatea de personal este realizată de către un compartiment specializat din cadrul unității economice, compartiment a cărui mărime și structură este dictată de volumul activităților desfășurate și de nivelul ierarhic al unității în care funcționează.

Pentru o unitate economică productivă acest compartiment va purta, în referirile noastre, numele de compartimentul de Personal, Învățămînt, Retribuirea muncii (P.I.R.).

În acest context compartimentul P.I.R. își desfășoară activitatea în următoarele domenii specifice :

— stabilirea și asigurarea necesarului de personal pe meserii, funcții și specialități corespunzător planului de producție și investiții ;
— pregătirea și calificarea forței de muncă ;
— perfecționarea, pregătirea și reciclarea personalului-muncitor ;
— aplicarea cu strictețe a reglementărilor legale privind remunerația muncii ;
— încadrarea în muncă ;
— promovarea muncitorilor calificați în categorii de calificare ;
— promovarea în funcții de conducere ;
— promovarea dintr-o funcție inferioară într-o funcție superioară (care nu este funcție de conducere) ;
— trecerea personalului în trepte sau gradații superioare de retribuire ;
— transferarea în interes de serviciu sau la cererea persoanelor încadrate în muncă ;
— schimbarea locului de muncă în cadrul aceleiași localități ;
— desfacerea contractului de muncă ;
— evidența personalului-muncitor ;
— evidența timpului de lucru ;
— programarea concediilor de odihnă.
— elaborarea situațiilor de informare-raportare cu privire la personalul-muncitor și utilizarea forței de muncă etc.

Relațiile interne și externe ale compartimentului P.I.R. — Fluxul informațional

Schematic diagrama sintetică a relațiilor compartimentului P.I.R. cu celelalte compartimente din unitatea economică și din afara acesteia se prezintă ca în figura 10.1.

Fluxul informațional stabilit prin legăturile evidențiate în figura 10.1 poate fi descris astfel :

1. Relațiile compartimentului P.I.R. cu centrala industrială.

a) Transmite centralei :

— deficitul și excedentul de personal-muncitor pe meserii și specialități ;
— propunerile pentru acordarea de retribuții tarifare majorate ;
— propuneri de completare a nomenclatorului de funcții ;
— proiectul planului de formarea și perfecționarea pregătirii profesionale a personalului-muncitor ;
— indicatorii specifici de realizarea cărora este condiționată acordarea retribuției tarifare etc. ;

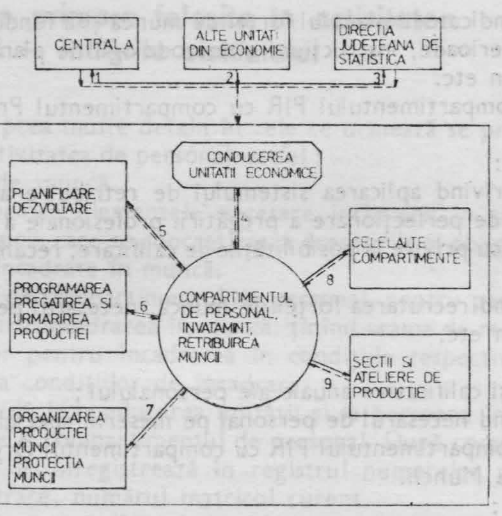


Fig. 10.1. Diagrama de relații a compartimentului de Personal, Învățămînt, Retribuire

b) Primește de la centrală :

– indicații, metodologii, norme și instrucțiuni pe probleme de personal, învățămînt, retribuirea muncii precum și aprobări privind locurile de muncă cu retribuția tarifară majorată ;

– planul de formare și perfecționare a pregătirii profesionale a personalului muncitor și de redistribuire a unor specialiști ;

– indicatorii specifici de realizarea cărora este condiționată acordarea retribuției tarifare etc.

2. Relațiile compartimentului P.I.R. cu alte unități economico-sociale.

a) Transmite către alte unități :

– documente privind transferările în interes de serviciu ;

– cereri de comunicare a antecedentelor penale pentru unele funcții etc. ;

b) Primește de la alte unități economico-sociale :

– documente privind transferările în interes de serviciu ;

– referințe pentru cadrele din nomenclatură ;

– comunicări privind antecedentele penale pentru unele funcții etc.

3. Relațiile compartimentului P.I.R. cu D.J.S.

a) Primește de la DJS metodologii și norme referitoare la indicatorii statistici ceruți.

b) Transmite raportări statistice.

4. Relațiile compartimentului PIR cu conducerea unității economice.

a) Transmite : informări și propuneri pentru îmbunătățirea activității compartimentului, indicatori și rapoarte nominale privind datele aferente personalului ;

b) Primește sarcini și indicații în legătură cu desfășurarea activității compartimentului, cereri de elaborare a unor raportări numerice sau nominale, în afara celor tipizate ;

5. Relațiile compartimentului PIR cu compartimentul Planificare-Dezvoltare.

a) Transmite date privind fundamentarea planului de muncă și a fondului de retribuire ;

b) Primeşte indicatorii planului forţei de muncă şi a fondului de retribuire pe compartimente şi perioade, instrucţiuni şi metodologii de planificare şi urmărire a indicatorilor de plan etc.

6. Relaţiile compartimentului PIR cu compartimentul Programarea şi Urmărirea Producţiei.

a) Transmite :

— indicaţii privind aplicarea sistemului de retribuire a muncii ;

— programe de perfecţionare a pregătirii profesionale a personalului ;

— informaţii cu privire la posibilităţile de calificare, recalificare şi policalificare a personalului.

— date privind recrutarea forţei de muncă, necesarul pe meserii, specialităţi şi perioade de plan etc.

b) Primeşte :

— aprecieri şi calificative anuale ale personalului ;

— date privind necesarul de personal pe meserii, specialităţi şi perioade etc.

7. Relaţiile compartimentului PIR cu compartimentul Organizarea Producţiei, Muncii şi Protecţia Muncii.

a) Transmite :

— lucrări necesare la elaborarea unor studii şi analize privind activitatea de producţie şi personal-muncitor ;

— programe de formare şi perfecţionare a pregătirii personalului etc.

b) Primeşte :

— indicatoarele tarifare de calificare ;

— norme şi normative de muncă ;

— criterii de gradare a întreprinderii, secţiilor, atelierelor, şantierelor etc. ;

— propuneri de programe pentru formarea şi perfecţionarea pregătirii profesionale a personalului în domeniul organizării producţiei şi a muncii ;

— lucrări în colaborare privind necesarul de personal etc.

8. Relaţiile compartimentului P.I.R. cu alte compartimente.

a) Transmite către alte compartimente :

— indicatori privind aplicarea sistemelor de retribuire a muncii ;

— programe de formare şi perfecţionare a pregătirii personalului ;

— propuneri de promovare şi sancţionare a personalului ;

— date privind fluctuaţia şi starea disciplinară a personalului etc.

b. Primeşte :

— propuneri pentru programele de formare şi perfecţionare a pregătirii profesionale ;

— aprecierile şi calificativele anuale ale personalului etc.

9. Relaţiile P.I.R. cu Secţiile şi Atelierele de Producţie se referă la primirea şi transmiterea unor date în legătură cu pregătirea profesională şi retribuirea personalului muncitor.

În cadrul activităţii de personal domeniul care vehiculează cel mai mare volum de date pe baza cărora se elaborează indicatorii necesari activităţii de personal-muncitor este cel de evidenţă a personalului.

Totodată, acest domeniu presupune şi cel mai mare volum de muncă de rutină, de constatare şi consemnare, rămînînd prea puţin timp pentru activitatea de analiză şi decizie privind activitatea de personal.

Din aceste considerente în aplicaţia noastră ne vom concentra atenţia asupra acestei probleme.

Documentele primare folosite în activitatea de personal – intrările sistemului

Fără a intra în prea multe detalii în cele ce urmează se prezintă principalele documente pentru activitatea de personal, astfel :

a) Contractul de muncă.

Formularul cuprinde elementele necesare încheierii în scris a contractului individual de muncă, prin care se concretizează drepturile și obligațiile reciproce ale unității și persoanei încadrate în muncă.

Se completează de compartimentul de personal, pentru persoanele care îndeplinesc condițiile pentru încadrarea în muncă, ținînd seama de rezultatele verificării îndeplinirii condițiilor pentru încadrarea în condițiile respective, consemnate în fișa pentru verificarea condițiilor de încadrare.

Formularul circulă la conducerea unității și la persoana încadrată în muncă, se păstrează la unitate la compartimentul de personal. După completare și semnare, contractul de muncă se înregistrează în registrul numerelor matricole, primind ca număr de înregistrare, numărul matricol curent.

b) Contractul de muncă (forma simplificată pentru perioade limitate de încadrare în muncă).

Pentru încadrările în muncă la lucrări cu caracter sezonier în agricultură sau ca personal nepermanent în construcții, întreținerea drumurilor, încărcări-descărcări și alte asemenea, se recomandă folosirea formularului „contract de muncă" într-o formă simplificată, pentru a se reduce la minimum volumul de muncă necesitat de astfel de cazuri, pentru operațiile de încadrare în muncă.

Pentru contractele de muncă de acest tip, nu se folosește fișa pentru verificarea condițiilor de încadrare și nu se înregistrează în registrul numerelor matricole.

c) Dispoziția de desfacere a contractului de muncă.

Se întocmește de compartimentul de personal, la indicația conducerii unității, arătîndu-se motivele și prevederile legale pe care se întemeiază desfacerea contractului de muncă, precum și organele la care măsura se poate ataca.

Dispoziția de desfacere a contractului de muncă poartă avizul oficiului juridic al unității.

d) Registrul numerelor matricole

Servește ca evidență multiplă a numerelor matricole acordate personalului unității, a carnetelor de muncă și a datei încheierii și încetării contractelor de muncă, înlocuind registrele, personalul intrat și ieșit din serviciu, precum și cel de evidență a carnetelor de muncă.

Se recomandă ca numărul matricol să fie cel al contractului de muncă. Fiecare persoană nou încadrată primește un număr matricol, care constituie codul de identificare în relațiile cu unitatea ; în cazul încetării contractului de muncă, numărul matricol deținut de fostul lucrător, nu se atribuie altei persoane nou încadrate.

e) Fișa personală

Servește la consemnarea pentru fiecare persoană încadrată în muncă, a datelor de identitate, stare civilă, studii, pregătire profesională etc.

Completarea fișei personale se face pe baza actelor originale prezentate de titular la încadrarea sa în muncă și pe parcursul executării contractului de muncă, ținîndu-se la zi, cu toate modificările ce intervin față de datele înscrise inițial.

Pe baza fișei, se completează și capitolul cu date personale din fișa de evidență a retribuțiilor, ce se transmite la încadrarea în muncă, compartimentului care întocmește statele de retribuții.

f) Registrul de evidență a mișcării personalului

Document de ieșire, servește pentru evidența la zi a efectivului scriptic pe total muncitori și total personal tehnic, economic, de altă specialitate și administrativ, urmărește modificările care au loc în totalul efectivului categoriei respective ca urmare a intrărilor și ieșirilor de personal. Totodată, servește ca evidență pe cauze a plecărilor din unitate în scopul analizei și determinării factorilor care provoacă mișcarea și fluctuația personalului.

Registrul se completează de compartimentul de personal pe total unitate, iar în caz de nevoie și pe sectoarele unde fluctuația este mai mare.

g) Situația utilizării timpului de lucru

Servește la evidențierea zilnică, la nivelul compartimentului, a numărului persoanelor prezente și absente din totalul efectivului scriptic al categoriei de personal respective, fiind un document de ieșire.

Situația din ziua curentă se completează cu datele care rezultă din documentele de prezență cu privire la numărul celor care au venit sau nu la lucru în ziua și schimbul respectiv.

Situația se întocmește pe fiecare schimb, pe muncitori și personal TESA.

h) Evidența persoanelor care îndeplinesc condițiile de trecere într-o gradație (treaptă) superioară de retribuire.

Situația de ieșire în care sînt centralizate persoanele care îndeplinesc condițiile cerute de lege de trecere într-o gradație (treaptă) superioară de retribuire, precum și a persoanelor care n-au vechimea necesară de trecere într-o gradație (treaptă superioară, dar care datorită unor realizări deosebite în muncă, li se reduce vechimea minimă. Situația este tipizată pe economie.

Cînd o persoană care a plecat din unitate este din nou încadrată, primește numărul matricol deținut anterior, iar în registru se înscrie noua dată a încheierii contractului de muncă. În situația cînd nu se poate identifica numărul matricol anterior, la încadrare se va acorda un număr matricol curent.

Necesitatea introducerii evidenței după numere matricole va fi hotărîtă de către organul de conducere colectivă sau de conducătorul unității, în funcție de specificul de organizare al unității.

La introducerea acestei evidențe, se vor acorda numere matricole în primul rînd personalului existent și contractelor de muncă încheiate pînă la data respectivă, după care fiecărui nou contract de muncă ce se va înregistra în registru, i se va da numărul matricol curent.

Registrul numerelor matricole se păstrează și se completează de compartimentul de personal al unității care efectuează încadrarea în muncă.

i) Carnetul de muncă

Carnetul de muncă constituie actul oficial prin care o persoană încadrată în muncă dovedește vechimea în muncă, vechimea neîntreruptă în muncă și în aceeași unitate, vechimea în funcție sau meserie, retribuția tarifară de încadrare, în vederea obținerii, în condițiile legii, a unor drepturi.

Carnetul de muncă se întocmește, se completează, se păstrează și se utilizează în conformitate cu prevederile Decretului nr. 92/1976 și Ordinul ministrului muncii, publicat în Buletinul Oficial nr. 76 din 29 iulie 1976.

Toate aceste documente sînt tipizate și standardizate sub aspectul conținutului informațional. În ceea ce privește forma de prezentare, acestea, pot fi modificate și proiectate (adaptate) în conformitate cu cerințele prelucrării automate a datelor. Astfel, este indicat ca datele ce urmează a fi preluate și transmise direct din documentele primare să fie încadrate cu linii discrete în „căsuțe". În măsura în care datele de intrare urmează a fi perforate în cartele, în dreptul fiecărei cifre a indicatorilor ce urmează a fi preluați se vor trece coloanele corespunzătoare din cartele. În acest mod se realizează, o corespondență între informațiile din document și pozițiile corespunzătoare din coloanele cartelei.

În ceea ce privește forma de aranjare a indicatorilor în formular (document), trebuie să se asigure o succesiune logică a datelor ce urmează a fi culese, transmise și prelucrate, respectînd următoarea ordine : în prima parte a formularului vor fi prevăzute datele de identificare, apoi cele de grupare și în final vor fi prevăzute rubricile cu date variabile cantitativ-valorice.

Proiectarea documentelor de intrare în funcție de aceste considerente oferă posibilitatea preîntîmpinării apariției erorilor în procesul de culegere, transmitere și prelucrare a datelor.

Situațiile de informare-raportare privind activitatea de personal — ieșirile sistemului

Se știe în ceea ce privește raportul dintre activitatea de bază din unitatea economică și sistemul informațional, că sistemul informațional reprezintă „umbra" tuturor activităților și proceselor economice, urmărind îndeaproape fluxul tehnologic din unitate, deci modul cum se succed valorile materiale în procesul de producție, cum se respectă legislația și normativele de utilizare a capacităților de producție, a forței de muncă etc.

Pentru buna funcționare a activităților din întreprindere, pentru atingerea obiectivului activității de bază, conducerea întreprinderii trebuie să cunoască în permanență modul concret în care se desfășoară activitatea și să intervină pentru înlăturarea unor eventuali factori perturbatori și limitarea consecințelor acestora. Pentru a putea face față acestor cerințe, sistemului informațional îi revine sarcina, pe de o parte, de a furniza conducerii informații necesare în scopul elaborării deciziilor ce se impun, iar pe de altă parte, de a transmite spre procesul de producție deciziile luate de conducere.

Conducerea formulează cerințe în ceea ce privește modul, conținutul și periodicitatea informațiilor (situațiilor de informare-raportare) ce urmează a fi furnizate prin sistemul informațional, iar calitatea informațiilor și modul lor de prelucrare va influența calitatea deciziilor și implicit eficiența întregii activități din întreprindere.

Plecînd de la o analiză a metodelor de conducere, precum și de la sarcinile conducerii unităților economice în general, apreciem că în formularea cerințelor față de sistem trebuie avute în vedere o serie de considerente, unele dintre ele constituind chiar principii ale informării, cum ar fi :

a) În definirea cerințelor (situațiilor de informare-raportare) trebuie să se aplice principiul selecției indicatorilor necesari fiecărui nivel de conducere și informării prin excepție. Este cunoscut faptul că un volum exagerat de informații nu reprezintă

situaţia cea mai fericită pentru conducere, o astfel de cantitate de informaţii generează practic o serie de dificultăţi ale căror consecinţe nu pot fi ignorate în conducerea operativă a întreprinderii. De aceea, cu ocazia stabilirii cerinţelor conducerii, trebuie să se definească toate elementele necesare aplicării principiului „selecţiei şi informării prin excepţie" în toate situaţiile posibile. A ţine seama de acest principiu înseamnă a degreva conducerea de o serie de informaţii ce nu sînt utile în procesul de conducere, înseamnă a veni în sprijinul acesteia.

Cantitatea exagerată de informaţii prezintă cel puţin următoarele aspecte negative :

— necesită un volum sporit de muncă ocazionat de culegerea, verificarea, transmiterea, prelucrarea şi stocarea datelor ;

— duce la sufocarea conducerii cu un volum de date inutile ceea ce va afecta extrem de mult operativitatea în procesul luării deciziilor ;

— excesul de date culese şi prelucrate în paralel, nu este de natură să sprijine conducerea, ci din contră poate să creeze dezorientare prin faptul că aceleaşi fenomene sau operaţii pot fi prezentate prin date diferite, uneori contradictorii, pentru lămurirea cărora este necesară o muncă suplimentară de verificare şi punere de acord. De aceea, volumul informaţiilor trebuie să se stabilească în funcţie de cerinţele reale ale fiecărui nivel ierarhic de conducere pe baza unei analize aprofundate a acestora şi selecţionarea lor atentă care să permită eliminarea celor inutile, nesemnificative sau redondante.

b) Stabilirea corespunzătoare a numărului de exemplare a situaţiilor de ieşire care se întocmesc, precum şi a frecvenţei de elaborare a acestora. Deşi această problemă reprezintă o consecinţă directă a aplicării principiului „selecţiei şi informării prin excepţie", totuşi datorită multiplelor implicaţii pe care le are va fi tratată separat.

Numărul de exemplare în care se elaborează situaţiile de ieşire, precum şi frecvenţa acestora, va influenţa volumul datelor şi documentelor ce vor circula în sistem, avînd diverse implicaţii şi nu lipsite de importanţă. Numărul exemplarelor reclamă un consum mai mare de hîrtie şi un volum de muncă suplimentar pentru multiplicare, difuzare, înregistrare, arhivare. Nu trebuie scăpat din vedere faptul că volumul mare de exemplare necesită de asemenea spaţii de păstrare în perioada curentă şi de arhivare după aceea, care presupun încăperi în plus cu toate cheltuielile de amenajare aferente. Totodată, în multe cazuri, numărul de exemplare nu este justificat integral de cerinţele conducerii sau compartimentelor prin care circulă, ceea ce face ca o bună parte din ele să nu aibă importanţă pentru compartimentele în care circulă. Ca atare ele încarcă inutil volumul de hîrtii, de lucrări precum şi munca personalului respectiv cu primirea, înregistrarea, îndosarierea acestora etc.

c) Gradul de prelucrare al informaţiilor trebuie să corespundă nivelului ierarhic şi să fie în concordanţă cu destinaţia şi scopul urmărit prin utilizarea acestor informaţii. Astfel, la nivel inferior de decizie, informaţiile vor avea un grad analitic, vor fi prezentate într-o formă mai „brută" (mai puţin prelucrate), iar la nivele superioare informaţiile trebuie să aibă un grad sporit de sintetizare, ceea ce implică un grad mai mare de prelucrare a datelor (figura 10.2). Totodată, nivelul de prelucrare a datelor trebuie corelat cu scopul în care sînt utilizate informaţiile, cu caracterul acestor informaţii. Astfel, informaţiile cu caracter operativ, furnizate zilnic sau săptămînal, vor fi de regulă neprelucrate sau vor prezenta un grad redus de prelucrare, dat fiind funcţia acestora de a sesiza cît mai operativ dereglările şi anomaliile ce se manifestă în activitate, pentru a da posibilitatea unor intervenţii prompte, cu maximă eficacitate.

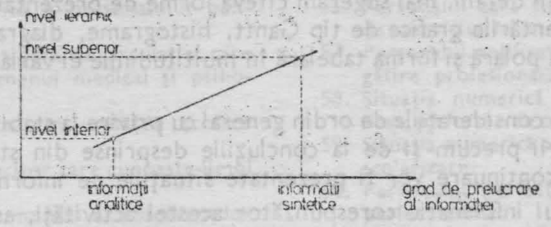


Fig. 10.2. Gradul de prelucrare a informaţiilor la diferite nivele de conducere

Din aceste considerente, informaţiile care se cer cunoscute zilnic trebuie alese cu mult discernămînt, pentru a exprima ceea ce este esenţial pentru conducerea curentă; în acelaşi timp să nu fie prea numeroase şi să solicite un volum prea mare de muncă prin care să periclitezе operativitatea.

Indicatorii care reprezintă o prelucrare mai complexă, pe bază de calcule şi corelaţii stabilite prin luarea în considerare a unui volum mai mare de date diferite, sînt furnizaţi, de obicei, la intervale de timp mai lungi, presupunînd un volum de muncă mai mare. Datorită faptului că sînt furnizaţi la intervale mai mari, aceşti indicatori au un grad de operativitate mai redus, dar în schimb au o valoare informativă mai ridicată întrucît prezintă fenomenele în legătura lor cauzală intimă şi permit analiza complexă a rezultatelor, aprecierea nivelului calitativ al deciziilor elaborate şi conturarea mai clară şi precisă a tendinţelor.

d) Indicatorii trebuie prezentaţi în forma evolutivă pentru a putea sugera o anumită tendinţă în desfăşurarea fenomenului. O informaţie necorelată cu un nivel anterior şi cu obiectivul planificat nu este semnificativă (fig. 10.3).

e) Forma de prezentare a indicatorilor trebuie să avertizeze operativ asupra evoluţiei necorespunzătoare a unui fenomen.

Totodată, în funcţie de limitele de evoluţie ale unui fenomen, informaţia poate servi la fundamentarea deciziei unui factor sau altul la anumite niveluri ierarhice. În cazul în care abaterile constatate devin mai mari decît un anumit nivel prestabilit se impune consultarea factorilor de decizie de la nivelul ierarhic superior, acestea fiind considerate abateri de gradul II. Un astfel de grafic de prezentare a abaterilor nivelului de informaţii este sugerat în fig. 10.4.

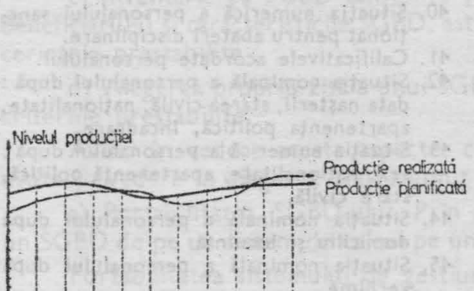


Fig. 10.3. Diagrama producţiei

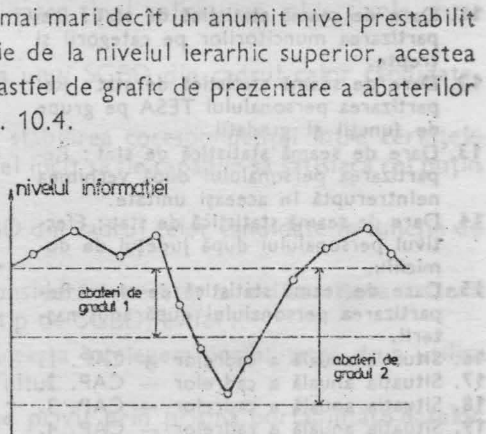


Fig. 10.4. Graficul abaterilor nivelurilor de informaţii

Fără a intra în detalii, mai sugerăm cîteva forme de prezentare a indicatorilor, cum ar fi: reprezentările grafice de tip Gantt, histograme, diagrama de structură circulară, diagrama polară și forma tabelară în multitudinile ei variante, fiind cea mai răspîndită.

Plecînd de la considerațiile de ordin general cu privire la stabilirea informațiilor necesare conducerii precum și de la concluziile desprinse din studiul „Activității de Personal", în continuare vor fi prezentate situațiile de informare-raportare ca ieșiri ale sistemului informatic corespunzător acestei activități, astfel:

1. Dare de seamă statistică centralizată: Efectivul personalului pe ramuri (M3).
2. Dare de seamă statistică de stat: Asigurarea muncitorilor calificați pe meserii.
3. Dare de seamă statistică de stat: Îndeplinirea planului de Școlarizare a muncitorilor calificați.
4. Dare de seamă statistică de stat: Asigurarea personalului cu pregătire medie și superioară.
5. Dare de seamă statistică de stat: Cheltuieli de Școlarizare.
6. Dare de seamă statistică de stat: Personalul care primește sporuri de retribuții.
7. Dare de seamă statistică interdepartamentală privind perfecționarea pregătirii profesionale.
8. Dare de seamă statistică departamentală privind mișcarea personalului, sancțiuni și navetiști.
9. Dare de seamă statistică de stat: Repartizarea personalului pe grupe de retribuții.
10. Dare de seamă statistică de stat: Personalul angajat pentru prima oară în anul ...
11. Dare de seamă statistică de stat: Repartizarea muncitorilor pe categorii și trepte.
12. Dare de seamă statistică de stat: Repartizarea personalului TESA pe grupe de funcții și gradații.
13. Dare de seamă statistică de stat: Repartizarea personalului după vechimea neintreruptă în aceeași unitate.
14. Dare de seamă statistică de stat: Efectivul personalului după județul de domiciliu.
15. Dare de seamă statistică de stat: Repartizarea personalului după anul nașterii.
16. Situația anuală a cadrelor — CAP. 1.
17. Situația anuală a cadrelor — CAP. 2.
18. Situația anuală a cadrelor — CAP. 3.
19. Situația anuală a cadrelor — CAP. 4.
20. Personalul cu obligații militare.
21. Tabel nominal cu tinerii ce urmează să recruteze.
22. Personalul promovat în funcție sau categorie, în condiții normale și cu reducere de stagii.
23. Raporturi între categorii de personal, pe activități.
24. Personalul de conducere directă a producției.
25. Personalul cu studii superioare.
26. Personalul cu studii medii.
27. Situația numerică a personalului cu studii superioare, pe funcții și specialități.
28. Situația numerică a personalului cu studii medii pe funcții și specialități.
29. Situația nominală a muncitorilor pe meserii, categorii de calificare și trepte de retribuție.
30. Situația numerică a personalului pe funcții/meserii, ponderea femeilor.
31. Personalul navetist: funcții din siguranța circulației.
32. Personalul navestit: funcții/meserii care nu sînt din siguranța circulației.
33. Personalul cu funcții de gestiune.
34. Personalul de pază, pompieri și deservire.
35. Personalul cu distincții și decorații.
36. Personalul recompensat.
37. Situația numerică a personalului recompensat.
38. Cadre de conducere eliberate din funcție.
39. Personalul sancționat pentru abateri disciplinare.
40. Situația numerică a personalului sancționat pentru abateri disciplinare.
41. Calificativele acordate personalului.
42. Situația nominală a personalului după: data nașterii, starea civilă, naționalitate, apartenența politică, încadrare.
43. Situația numerică a personalului după: sex, naționalitate, apartenență politică, stare civilă.
44. Situația nominală a personalului după domiciliu și locuință.
45. Situația nominală a personalului după vechime.
46. Personalul ce urmează a se pensiona.
47. Situația numerică a personalului ce urmează a se pensiona.

48. Situaţia numerică a personalului pensionat.
49. Personalul din siguranţa circulaţiei care a efectuat examenul medical şi psihotehnic.
50. Cadre didactice din unităţile şcolare subordonate.
51. Personalul muncitor care cunoaşte limbi străine.
52. Încadrarea în muncă a absolvenţilor liceelor de specialitate. Şcoli maiştri şi profesionale.
53. Încadrarea în muncă a absolvenţilor învăţămîntului superior.
54. Stagiarii care au susţinut examenul după primul an de activitate.
55. Stagiarii care au susţint lucrarea practică pentru definitivarea în funcţie.
56. Personalul care s-a deplasat în interes de serviciu în străinătate.
57. Personalul calificat, pe forme de pregătire profesională.
58. Situaţia numerică a personalului calificat, pe forme de pregătire profesională.
59. Situaţia numerică a realizării planului de şcolarizare.
60. Personalul perfecţionat şi în curs de perfecţionare.
61. Situaţia numerică a personalului aflat în curs de perfecţionare profesională.
62. Cadre de conducere perfecţionate şi în curs de perfecţionare.
63. Personalul care îndeplineşte condiţiile pentru acordarea de gradaţii şi trepte.
64. Personalul care îndeplineştşe condiţiile pentru schimbarea sporului de vechime.

Pe lîngă aceste cerinţe informaţionale mai pot exista şi altele, însă ele reprezintă doar editări de situaţii după alte criterii (chei) de grupare, fără să difere prea mult sub aspectul conţinutului informaţional faţă de cele prezentate anterior.

Prin satisfacerea acestor cerinţe apreciem că sistemul informatic ar putea constitui un adevărat instrument la dispoziţia conducerii unităţii economice, degrevînd personalul-muncitor de o serie de activităţi de rutină şi asigurînd exactitate şi operativitate în activitatea de raportare-informare.

Alegerea sistemului de gestiune – De ce SGBD-SOCRATE ?

Procesul de alegere a unui SGBD presupune următoarele faze :

a) stabilirea cerinţelor beneficiarului de sistem şi studiul acestora sub aspectul tipurilor de structuri de date ce le implică, timpul de răspuns pentru cerinţele respective, metodelor de acces, confidenţialitate, tipul aplicaţiilor, obiectivele sistemului etc. ;

b) Stabilirea criteriilor de alegere a unui SGBD din cadrul celor candidate, în funcţie de cerinţele beneficiarului ;

c) Inventarierea SGBD existente şi stabilirea corespondenţei între cerinţele beneficiarului şi caracteristicile SGBD, astfel încît să fie capabile să satisfacă cel puţin cerinţele prestabilite ;

d) Alegerea propriu-zisă a unui SGBD din cadrul celor candidate în funcţie de criteriile prestabilite.

Avînd în vedere aceste aspecte, considerăm necesar a indica cîteva criterii avute în vedere în alegerea unui anumit tip de SGBD, astfel :

a) Portabilitatea SGBD-ului. Prin aceasta înţelegem posibilitatea de a utiliza un SGBD de pe un sistem de calcul pe un altul.

Portabilitatea sistemului de gestiune privit prin prisma portabilităţii datelor se referă la posibilitatea de a folosi o serie de date utilizate în cadrul unui sistem informatic de către un alt sistem informatic, deci posibilitatea integrării fişierelor deja existente în cadrul unui alt sistem ;

b) Costul sistemului. Acest criteriu trebuie privit prin prisma :
— timpului de ocupare a unității centrale ;
— costului de întreținere și dezvoltare ;
— resurselor hardware imobilizate ;
— costului de adaptare și trecere pe un nou sistem de calcul ;
— costul documentației etc.

c) Facilitățile de implementare, întreținere și exploatare a băncii de date. Acestea sînt reflectate prin :
— modalitatea de descriere a datelor ;
— tehnicile de organizare și regăsire a datelor, care să permită un acces cît mai rapid la orice informație ;
— timpul cît mai redus pentru actualizare, căutare și răspuns la cererile de informare ;
— editarea operativă a celor mai variate tipuri de situații solicitate de către utilizator ;
— posibilitatea de inserție a unor programe de aplicație, programe de validare de date, de actualizare, rutine statistice, rutine de sortare, rutine de prezentare grafică a ieșirilor etc. ;

d) Posibilitatea gestionării structurilor complexe de date, cum ar fi cele de tip arborescent sau rețea ;

e) Multitudinea metodelor de acces. În funcție de cerințele proprii aplicației, sistemul va trebui să suporte interogări sau actualizări în „timp real" avînd proceduri de tip conversațional ;

f) Protecția și securitatea datelor din bază ;

g) Specificul aplicației. E cunoscut că programele sînt orientate pe aplicații, cum ar fi : programarea producției, aprovizionare-desfacere, optimizări, prognoze etc. ;

h) Timpul necesar pentru formarea cadrelor care să utilizeze SGBD etc.

Toate aceste criterii de alegere pot fi corelate cu o serie de factori complementari cum ar fi : mentenanța sistemului, facilitățile ce le oferă administratorului băncii de date, calitatea documentației oferită de furnizor, asistență în implementarea sistemului și în pregătirea cadrelor etc.

Toți acești factori alături de criteriile enunțate pot să influențeze succesul în implementarea SGBD și eficiența economică pe ansamblul sistemului informatic.

În acest context, considerentele pentru care optăm pentru SGBD-SOCRATE sînt :

— structurile de date generate de aplicațiile referitoare la activitatea de personal sînt de tip arborescent și rețea, structuri care sînt tratate de SGBD-SOCRATE ;

— asigură confidențialitatea și integritatea datelor din bază ;

— sub aspectul portabilității SGBD, portabilitatea se rezolvă de la sine, întrucît economia națională este dotată aproape în totalitate cu sisteme de calcul din familia FELIX. Totodată, asigurînd interfețe cu limbajele de nivel înalt, oferă și posibilitatea integrării fișierelor deja existente în cadrul unui alt sistem, deci, în acest mod poate fi rezolvată și problema portabilității datelor ;

— SGBD-SOCRATE oferă mai multe tipuri de acces la date : secvențial (parcurgere naturală, parcurgere inversă, parcurgere completă), direct (prin număr de ordine, prin dicționar) și secvențial indexat ;

— SGBD-SOCRATE se livrează de către Întreprinderea de Calculatoare odată cu sistemul de calcul, fiind inclus ca o componentă în costul acestuia ;

— Oferă posibilitatea exploatării acestuia atît în mod batch cît şi conversaţional ;

— Pentru formarea şi instruirea cadrelor se organizează în mod sistematic cursuri de SOCRATE în cadrul Institutului de Cercetare Ştiinţifică şi Inginerie Tehnologică pentru Tehnică de Calcul şi Informatică — Bucureşti.

De ce bază de date pentru activitatea de PERSONAL ?

Considerentele pentru care optăm pentru o bază de date privind activitatea de personal sînt :

— Existenţa unui volum mare de date ce se vehiculează în acest domeniu de activitate. Există multe unităţi economice în care numărul personalului-muncitor este de peste 10 000, iar cunoaşterea informaţiilor despre o astfel de colectivitate şi conducerea acesteia presupune mari eforturi. În condiţiile unei baze de date această problemă se simplifică mult ;

— Datele referitoare la o persoană au în general, o mare stabilitate, ceea ce va uşura activitatea de actualizare a acestora ;

— Oferă posibilitatea accesului la date după o multitudine de chei, spre deosebire de organizarea datelor în fişiere, unde accesul are loc după o singură cheie ;

— Bazele de date oferă posibilitatea definirii unei multitudini de legături între entităţi. Acest lucru permite reducerea redundanţei datelor din bază cu multiplele avantaje ce pot fi obţinute în acest mod ;

— Sistemul rămîne deschis la modificările cadrului legal de funcţionare a sistemului ;

— Sistemele de gestiune a bazelor de date oferă limbaje de manipulare cu mari facilităţi în exploatarea interactivă a datelor.

Volumul, stabilitatea datelor precum şi multitudinea cheilor de acces la date sînt criterii hotărîtoare în opţiunea de organizarea datelor în baze de date.

Proiectarea structurii conceptuale a bazei de date

Din punct de vedere teoretic, proiectarea structurii bazei de date se încadrează în metodologia generală de realizare a sistemelor informatice, constituind o activitate din cadrul acesteia. În acest context, în activitatea de proiectare a structurii conceptuale a bazei de date se va ţine seama de cerinţele de informare ale conducerii (ieşirile sistemului) şi de informaţiile din documentele primare (intrările sistemului) identificate în baza de studiu şi analiză a sistemului informaţional existent.

În activitatea de proiectare a structurii conceptuale a bazei de date poate fi adoptată una din următoarele variante :

— proiectarea structurii bazei de date plecînd de la cunoaşterea cerinţelor informaţionale ale conducerii (ieşirile sistemului) spre intrările sistemului (metoda ieşiri-intrări) ;

— proiectarea structurii bazei de date plecînd de la intrările sistemului spre ieșirile sistemului (metoda intrări-ieșiri).

În primul caz structura conceptuală a bazei de date va fi dată de informațiile de intrare strict necesare pentru satisfacerea cerințelor prestabilite ale conducerii.

Avantajele metodei ieșiri-intrări constau în :

— culegerea, verificarea, transmiterea și stocarea unui volum redus de date, ceea ce va determina antrenarea unor resurse mai reduse de :

- operatori pentru culegerea, verificarea și transmiterea datelor ;
- echipamente de culegere, verificare, transmitere a datelor ;
- purtători tehnici de informații (benzi magnetice, discuri magnetice etc.) ;
- timp calculator necesar pentru încărcarea, întreținerea și prelucrarea datelor.

Dezavantajul metodei ieșiri-intrări derivă din faptul că baza de date nu conține multitudinea informațiilor de intrare din documentele primare decît cele apreciate strict necesare pentru satisfacerea cerințelor prestabilite ale conducerii, ceea ce înseamnă că există pericolul de a nu putea satisface și alte cerințe informaționale ale conducerii care fie că au scăpat formulării inițiale, fie că au apărut noi cerințe neprevăzute. Acest lucru poate determina reproiectarea structurii bazei de date.

În conformitate cu metoda intrări-ieșiri, structura bazei de date va fi dată de multitudinea informațiilor de intrare din documentele primare.

Avantajul acestei metode derivă din faptul că avînd stocate multitudinea datelor de intrare apare posibilitatea satisfacerii oricăror cerințe informaționale, chiar dacă acestea nu au fost prestabilite.

Dezavantajul metodei constă în faptul că se vehiculează cu volume mari de date, ceea ce implică un volum sporit de resurse umane, materiale și timp, ocazionate de culegerea, verificarea, transmiterea, stocarea și prelucrarea datelor.

Indiferent care dintre metode va fi adoptată structura bazei de date va fi dată de mulțimea, mai mult sau mai puțin extinsă, a identificatorilor datelor de intrare ce urmează a fi stocate. Mulțimea identificatorilor datelor de intrare poate fi privită ca o *„înregistrare universală"*. În cadrul „înregistrării universale" informațiile pot fi grupate în funcție de caracterul semantic al acestora într-o serie de entități (colecții), aplicînd tehnica analizei dependenței funcționale dintre identificatori (atribute) [ISCA85], [DATE82, 87], [DRĂGHICI], [COJOCARU].

În acest mod, pentru aplicația privind conducerea activității de personal au fost deduse următoarele entități : PERSOANA, CALIFICATIVE, PROFESII, FUNCȚII, SPECIALITĂȚI, COPII, JUDEȚE, LOCALITĂȚI, GRADE-MILITARE, NAȚIONALITATE-CETĂȚENIE, SANCȚIUNE, SANCȚIUNI, RECOMPENSE, RECOMPENSA.

Totodată, cerințele de ordin practic au impus și definirea următoarelor fișiere inverse : MUNCITORI, TESA, BĂRBAȚI și FEMEI asociate entității inversate PERSOANA. Pentru fiecare entitate se face o prezentare descriptivă corespunzătoare, care să pună în evidență o serie de aspecte cum ar fi : identificatorul entității, numărul maxim de realizări, structura fiecărei entități (caracteristicile acesteia), tipul fiecărei caracte-

ristici, restricții pentru valorile asociate identificatorilor, definirea cheilor de regăsire, eventualele legături dintre entitățile bazei de date etc.

În continuare vor fi deduse legăturile (relațiile) dintre entitățile bazei de date. În general legăturile dintre entitățile bazei de date pot fi de următoarele tipuri : „1" la „1" ; „1" la „N" (unul la mulți) și „M" la „N" (mulți la mulți).

Pentru baza de date referitoare la activitatea de PERSONAL relațiile dintre entități apar astfel :

– relațiile de tipul „1" la „1" sînt utilizate pentru a defini legături de forma SOT-SOTIE în cadrul entității PERSOANA ; COPIL-PĂRINTE etc. ;

– relațiile de tipul „1" la „M" arată că unei realizări dintr-o entitate A îi corespund mai multe realizări dintr-o altă entitate B (deci o structură arborescentă) și vor fi marcate printr-o săgeată (→), astfel :

- PERSOANA → COPII (o persoană poate avea mai mulți copii) ;
- PROFESII → PERSOANA (care sînt persoanele ce au aceiași profesie ; ce profesie are o anumită persoană) ;
- LOCALITĂȚI → PERSOANA (care sînt persoanele dintr-o anumită localitate) ;
- FUNCȚII → PERSOANA (care sînt persoanele ce au o aceeași funcție) ;
- SPECIALITATE → PERSOANA (care sînt persoanele ce au o aceeași specialitate) ;

– relații de tipul „M" la „N" (definesc structuri de tip rețea) și arată că unei realizări din entitatea A îi corespund mai multe realizări din entitatea B și invers, unei realizări din entitatea B pot să-i corespundă una sau mai multe realizări din entitatea A. Acest tip de relații va fi sugerat prin săgeți într-un sens și în altul ↔), astfel :

- PERSOANA ↔ SANCȚIUNE (o persoană poate avea una sau mai multe sancțiuni, o aceeași sancțiune poate fi dată unei singure persoane sau mai multor persoane ;
- PERSOANA – RECOMPENSA (o persoană poate beneficia de una sau mai multe recompense, o anumită recompensă poate fi acordată unei singure persoane sau mai multor persoane.

Tratarea relațiilor de tipul „M" la „N", presupune conversia acestora în relațiile de tipul „1" la „M", în scopul reducerii complexității programelor de încărcare, actualizare și exploatare a datelor corespunzătoare acestor structuri. În mod practic această conversie se realizează prin interpunerea unei entități de legătură între cele două entități ce definesc relația „M" la „N".

Pentru exemplul nostru relațiile de tipul „M" la „N" dintre entitățile PERSOANA și SANCȚIUNE au fost transformate în relații 1 la „M" prin interpunerea entității SANCȚIUNI (fig. 10.5).

Fig. 10.5. Descompunerea unei relații de tipul M la N în relații I la N

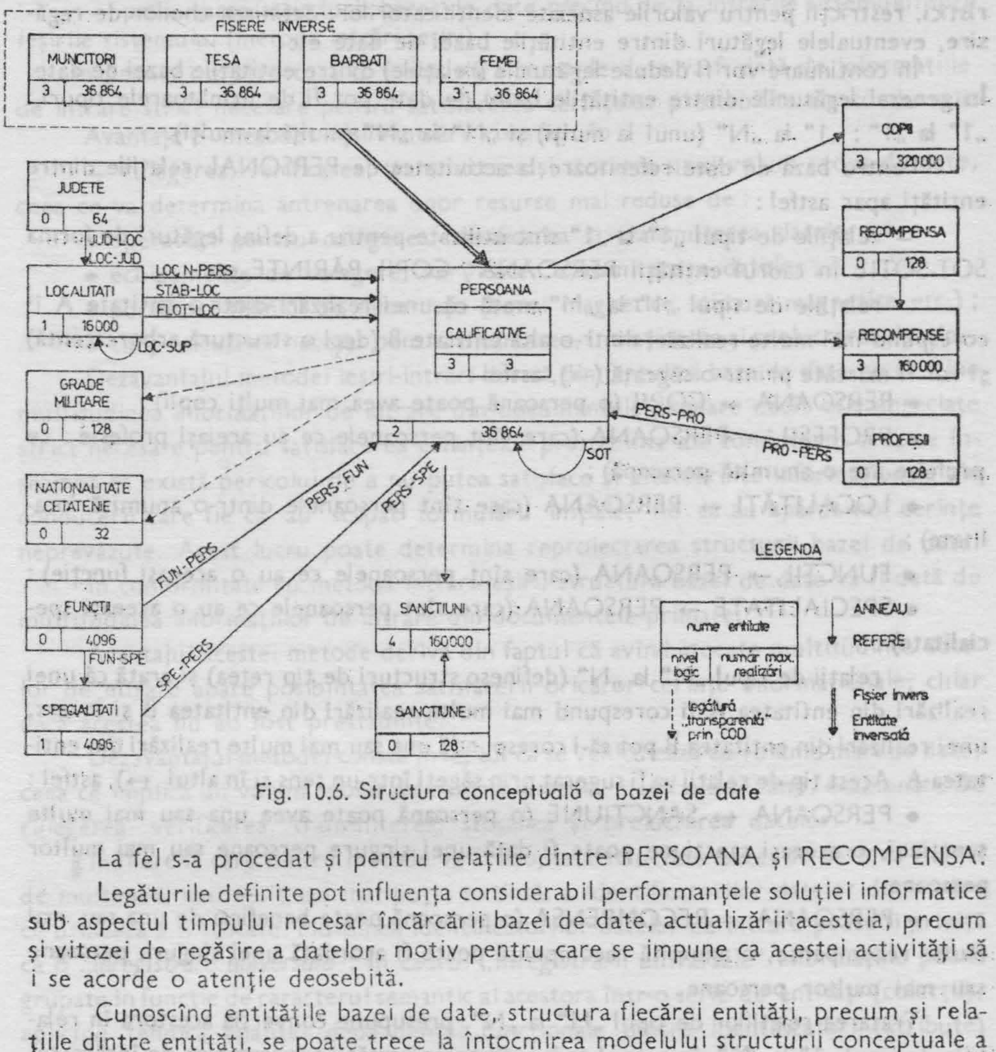


Fig. 10.6. Structura conceptuală a bazei de date

La fel s-a procedat şi pentru relaţiile dintre PERSOANA şi RECOMPENSA.

Legăturile definite pot influenţa considerabil performanţele soluţiei informatice sub aspectul timpului necesar încărcării bazei de date, actualizării acesteia, precum şi vitezei de regăsire a datelor, motiv pentru care se impune ca acestei activităţi să i se acorde o atenţie deosebită.

Cunoscînd entităţile bazei de date, structura fiecărei entităţi, precum şi relaţiile dintre entităţi, se poate trece la întocmirea modelului structurii conceptuale a bazei de date, aşa cum este sugerat în figura 10.6.

Odată elaborat modelul se poate trece la testarea structurii conceptuale a bazei de date. Prin această testare se urmăreşte să se vadă în ce măsură structura virtuală concepută este în măsură să satisfacă cerinţele conducerii specificate în paşii anteriori, dacă relaţiile dintre entităţi sînt corespunzător stabilite în vederea regăsirii, informaţiilor din mai multe entităţi şi editării unor situaţii complexe cu date agregate, dacă modul în care au fost definite legăturile dintre entităţi asigură coerenţa informaţiilor din bază etc. Această testare poate fi asemănată cu parte din activitatea de programare şi anume, odată formulată problema se trece la elaborarea şi testarea schemei logice de program, apoi se scrie programul corespunzător.

Fiind elaborat şi testat modelul corespunzător structurii conceptuale a bazei de date se poate trece la realizarea şi implementarea propriu-zisă a bazei de date utilizînd SGBD-SOCRATE.

11. IMPLEMENTAREA BAZEI DE DATE

GENERAREA BAZEI DE BAZE

Asa cum s-a specificat in capitolul 6 aceasta etapa se executa o singura data pentru mai multe baze de date (in exemplul nostru maxim 100). In cazul in care baza de baze exista generata aceasta etapa poate lipsi, utilizatorii urmind sa se conecteze la aceasta prin operatia de generare a bazei de date. Executia acestei etape, precum si gestiunea bazei de baze rezultate este o sarcina care revine administratorului bazei de baze. Acesta va defini protocolul de conectare, pentru o baza de date oarecare.

In acest exemplu de generare este apelata procedura generala PROCGEN pentru instalarea modulului SOCRATE-SOCGBDB al carui nume este furnizat ca parametru prin atribuire la identificatorul formal MODUL. Suprtul bibliotecilor cu modulele SOCRATE (parametrul SUPB) este discul sistem.

Baza de baze , numita "BAZA DE BAZE" , are dimensiunea fizica de 12 cilindri si este divizata logic (liniile 2-4) astfel:

- -FICH: 7 cilindri din care au rezultat 839 pagini cu dimensiunea subpagini de 16 cuvinte (linia 2);
- -DICO: 1 cilinru din care au rezultat 113 pagini cu dimensiunea subpagini de 4 (cuvinte);
- -PROG: 4 cilindri din care au rezultat 479 pagini cu dimensiunea subpagini de 256 cuvinte (linia 4).

Acesti parametri furnizati pe liniile 2-4 se pot modifica de catre administratorul in functie de necesitatile sale (spatiul real alocat ,in general).

Urmeaza descrierea structurii conceptuale a bazei de baze (liniile 5-60) si descrierea formalelor (liniile 61-96) pentru intrarile programelor de serviciu cu care este inzestrata baza de baze (liniile 99-423).

```
.        JOB PERSONAL,AN:1010,PN:SOC
.        * ALOCARE SPATIU PENTRU <BAZA DE BAZE>
.        ALLOC DVT:AD,VS:AD1AD1,AM:ANY,SZ:12,FN:'BAZA DE BAZE'
.        * ALOCARE SPATIU PENTRU <FISIER DE LUCRU SOCRATE>
.        ALLOC DVT:AD,VS:AD1AD1,AM:ANY,SZ:12,FN:'MANEVRA SGBD'
.        * GENERARE BAZA DE BAZE
.        XPROC PROCGEN,MOD
.        MOD &JURNAL:* &
.        MOD &MODUL:SOCGBDB&
.        MOD &SUPB:DV:AD0&
.        ENDMOD
.        FETCH LN:BIBIMTSO,GN:1,VN:1,FN:SOCGBDB,                  *      1
.              DV:AD0                                                   2
.        ASSIGN S,DVT:AD,VS:AD1AD1                                      3
.        ASSIGN Q,DVT:AD,VS:AD1AD1                                      4
.        LABEL S,AM:OFL,FN:'BAZA DE BAZE'                               5
.        LABEL Q,AM:OFL,FN:'MANEVRA SGBD'                               6
.        * ASSIGN Z,DV:MT00                                             7
.        * LABEL Z,FN:'JURNAL'                                          8
.        ASSIGN T,DV:AD7                                                9
.        * APPROCHE FIN PROCEDURE CATALOGUEE
.        EST                         FIN DE LA PROCEDURE CATALOGUEE
.        OPTION CS'CF:20,DS:4096'
.        RUN TIME:999,NL:500000,AD:0,0
socgbdb  started
nb cadre fich: 20
1    ¤
2    839     16        formatage bdb new-look
3    113     4
4    479     256
5    debut
6       entite  100  base
7          debut
8             nom mot 7
9             beb debut
10               org-struc de 0 a 1999999999
11                   org-fich  de 0 a 1999999999
```

```
                    org-mac   de 0 a 1999999999
                    entite 10 xbeb
                      debut
                        type ( 10 4 ) ( fich dico prog  stru )
                        nbpag       de 0 a 9999999
                        tspag       de 0 a 256
                        longsp      de 0 a 9999999
                        lgchbit     de 0 a 9999999
                        dep         de 0 a 9999999
                        sl11        de 0 a 9999999
                        sllpp2      de 0 a 9999999
                        srlpp2      de 0 a 9999999
                        sl132mk     de 0 a 9999999
                        sr132mk     de 0 a 9999999
                        pp2mask     de 0 a 512
                        kmask       de -1999999999 a -13
                        orgrel      de 0 a 9999999
            nb-cyl                  de 0 a 9999999
                      fin
              fin
          support debut
          type ( 9 6 ) ( md17 md25 md50 md100 )
                    entite 20 volume
                      debut
                        nom-proprio  mot 14
                        nb-cyl de 0 a 2000
                      fin
                    fin
                    entite  50  utilisateur
                      debut
                        nom       mot 15
                        numero  de 0 a 9999
                        mot-de-passe mot 7
            droit-utilisateur
              debut
                acces-macro  ( 15 12 ) /* sur 1 demi-octet */
                             ( imt comp imt+comp mac imt+mac
                             comp+mac imt+comp+mac restreint)
                acces-langage ( 15 11 )  /* 1 demi-octet */
                              ( autorise mise-a-jour
                              creation definition )
        pw-utilisateur de 1 a 16777215 /* 3 octets*/
              fin
                    fin
      fin
      pw1  mot  20
      pw2  mot  20
      pw3  mot  20
      pw4  mot  20
  formal carte-vol      debut
                        ident mot (3)
                        bid1 mot (1)
                        type mot (5)
                        bid2 mot (1)
                        nom-proprio mot 14
                        bid3 mot (1)
                        nb-cyl dilate (3)
                        bid4 mot (52)
                        fin
  formal carte-uti debut
                        ident mot (3)
                        bid1 mot (1)
                        nom mot (15)
                        bid2 mot (1)
                        numero dilate (4)
                        bid3 mot (1)
```

```
                         mot-de-passe mot (7)
                         bid4 mot (1)
                         acces-macro mot (12)
                         bid5 mot (1)
                         acces-langage mot (11)
                         bid6 mot (1)
                         pw-utilisateur dilate (8)
                         bid7 mot (14)
                         fin
                 formal carte-org  debut
                         ident  mot 3
                         bid1   mot 1
                         org-struc  dilate 10
                         bid2   mot 1
                         org-fich   dilate 10
                         bid3   mot 1
                         org-mac    dilate 10
                         bid4   mot 44
                         fin
     fin
     ?
!typ !identificateur                !cmax      !tail         !
!----!------------------------------!----------!-------------!
!bloc!fichier                       ! 1        ! 65429       !
!----!------------------------------!----------!-------------!
!enti!base                          ! 100      ! 654         !
!----!------------------------------!----------!-------------!
!form!carte-vol                     ! 1        ! 80          !
!----!------------------------------!----------!-------------!
!form!carte-uti                     ! 1        ! 80          !
!----!------------------------------!----------!-------------!
!form!carte-org                     ! 1        ! 80          !
!----!------------------------------!----------!-------------!
       |defimt    entreepw    ?
     entreepw 129/03/73088                             0000
     E!                                                0001
     entreepw                                          0002
     entreepw29/03/73088 001                           0003
     D358BC GGGGGG                                     0004
ok prog imt enregistre : entreepw
     :defpro sys1
     :exp
     m y1 = 0 g une base x1 m y1 = numde x1
     pour x1
          m nom = z1
          m org-struc de beb = 256
          m org-fich  de beb = 65536
          m org-mac   de beb = 1879048192
     fin
     d x6 = un volume de support de une base
     m x4 = formal carte-vol dans buf 1
     m y2 = 0
     faire
                 lire dans buf 1
                         m z6 = ident de x4
                         si z6 = 'vol'
                            alors g un volume x6 de support
                                   de x1
     m y2 = y2 + 1
                                  m type de support de x1 =
                                   type de x4
                            sinon sortie fin
                         pour x6
     m nom-proprio = nom-proprio de x4
                            m nb-cyl = nb-cyl de x4
                         fin
     refaire
```

```
    fin
        si y2 = 0  alors  i 'carte vol absente' s x1 fir
    d x6 = un utilisateur de une base
    m x5 = formal carte-uti dans buf 1
    m y2 = 0
    faire
                         m z6 = ident de x5
                         si z6 = 'uti'
                            alors pour x1 g un utilisateur x6
    m y2 = y2 + 1
                              fin
                            sinon sortie fin
                         pour x6
                            m nom = nom de x5
                            m numero = numero de x5
                            m mot-de-passe = mot-de-passe de x5
                            m acces-macro de droit-utilisateur
                              = acces-macro de x5
                            m acces-langage de
                              droit-utilisateur
                              = acces-langage de x5
                            m pw-utilisateur de
                              droit-utilisateur
                              = pw-utilisateur de x5
                         fin
                lire dans buf 1
    refaire
    fin
        si y2 = 0  alors  i 'carte uti absente' s x1 fin
    d x6
    m x6 = formal carte-org dans buf 1
    m z6 = ident de x6
    si z6 = 'org' alors
    pour x1
    m org-struc de beb =
    org-struc de x6
    m org-fich de beb =
    org-fich de x6
    m org-mac de beb =
    org-mac de x6
    fin
    fin
    m y17 = 0  /* initialise orgrel */
    m y23 = numde type de support de x1
    m y24 = d tout volume de support de x1
    m z7 = type de support de x1
    m y20 = 4
    m y21 = 120
    si z7 = 'diam' ou z7 = 'md17' alors
    m y20 = 1 m y21 = 30 fin
    m y19 = y21 * 30
    :fdef
    ?
ok prog comp enregistre  sys1
    :defpro sys2 :contxt d x1 = une base
    :exp
    g un xbeb x2 y16 de beb de x1
    pour x2
        si y15 = 0 alors m y2 = 0 s x2 sortie fin
    m y14 = y17
        m tspag  = y3
        m orgrel = y14
        m type   = z5
        m nb-cyl = y15
        m nbpag = y2     i type    i nbpag
        m y17   = y17 + y15 * y21 * 4
    fin
```

```
    :fdef
    ?
ok prog comp enregistre  sys2
    :defpro sys3
    :exp
    m y1 = 0 pour une base x1 ayant nom = z1 ;
           m y1 = numde x1
    m y17 = org-struc de beb de x1
    m y18 = org-fich de beb de x1
    m y19 = org-mac de beb de x1
    m z4 = nom-proprio de un volume de support de x1
    fin
    :fdef
    ?
ok prog comp enregistre  sys3
    :defpro sys4
    :exp
    m y2 = 0 m x1 = une base y1
    pour un utilisateur x2
         ayant nom = z2 et mot-de-passe = z3 ; de x1
           m y2 = numde x2
              m y3 = numde acces-macro de droit-utilisateur
                     de x2
              m y4 = numde acces-langage de droit-utilisateur
                     de x2
              m y5 = pw-utilisateur de droit-utilisateur de x2
              exec  entreepw
         fin
    :fdef
    ?
ok prog comp enregistre  sys4
    :defpro sys5
    :exp
    m y2 = 0 si z2 = pw4
      ou z2 = pw3
        alors m y2 = 1 fin
    :fdef
    ?
ok prog comp enregistre  sys5
    :defpro sys6
    :exp
    m y2 = 0 si z2 = pw4
      ou z2 = pw3
        ou z2 = pw2
        alors m y2 = 1 fin
    :fdef
    ?
ok prog comp enregistre  sys6
    :defpro sys7 :contxt d x1 = une base
    :exp
    s x1
    :fdef
    ?
ok prog comp enregistre  sys7
    :defpro sys8 :contxt d x1 = une base
    :exp
    m y2 = 0 /* pour le cas ou x2 indefini */
    pour un xbeb x2 y16 de beb de x1
       m y1  = numde x1
       m y2  = nbpag    de x2
       m y3  = tspag    de x2
       m y4  = longsp   de x2
       m y5  = lgchbit  de x2
       m y6  = dep      de x2
       m y7  = slll     de x2
       m y8  = sllpp2   de x2
       m y9  = srlpp2   de x2
```

```
        m y10 = sll32mk   de x2
        m y11 = srl32mk   de x2
        m y12 = pp2mask   de x2
        m y13 = kmask     de x2
        m y14 = orgrel    de x2
        m y23 = numde type de support de x1
        m y24 * d tout volume de support de x1
    fin
    :fdef
    ?
ok prog comp enregistre  sys8
    :defpro sys9 :contxt d x2 = un xbeb de beb de une base
    :exp
    pour x2
          m nbpag    = y2
          m tspag    = y3
          m longsp   = y4
          m lgchbit  = y5
          m dep      = y6
          m slll     = y7
          m sllpp2   = y8
          m srlpp2   = y9
          m sll32mk  = y10
          m srl32mk  = y11
          m pp2mask  = y12
          m kmask    = y13
    fin
    :fdef
    ?
ok prog comp enregistre  sys9
    :defpro sys10
    :exp
    m y2 = 0 si z2 = pw4
      alors m y2 = 1 fin
    :fdef
    ?
ok prog comp enregistre  sys10
      ]defmac  edibase  ]  ]  ]  ]
  ]exp   m z1 =   ']4]'
          si z1 = pw1 ou z1 = pw2 ou z1 = pw3 ou z1 = pw4
          alors
           m x1 = une base ayant nom = ']1]'
             et existe un utilisateur ayant nom = ']2]'
                          et mot-de-passe = ']3]' ; ;
             si existe x1 alors i x1
        sinon i (1) 'base*utilisateur*mot-de-passe'
                i (+2) 'inconnu ...' ecrire fin
          sinon i (1) 'mot de passe incorrect'
                i (+2) 'requete interdite ...' ecrire fin
      ]fdef ?
ok  macro  enregistree : edibase
       ]defmac   creation-util  de la base  ]  ]  ]  ]
    ]exp    m z1  =  ']4]'
        si z1 = pw2 ou z1 = pw3 ou z1 = pw4 alors
          m x1 = une base ayant nom = ']1]'
               et existe un utilisateur ayant nom = ']2]'
                          et mot-de-passe = ']3]' ; ;
             si  existe  x1  alors
                c un utilisateur  de x1
          sinon i (1) 'base*utilisateur*mot-de-passe'
                i (+2) 'inconnu ...' ecrire fin
           sinon i (1) 'mot de passe incorrect'
                i (+2) 'requete interdite ...' ecrire fin
    ]fdef  ?
ok  macro  enregistree : creation-util
      ]defmac   creabtch-util  de la base  ]  ]  ]  ]
   ]exp    m z1  =  ']4]'
```

```
           si z1 = pw2 ou z1 = pw3 ou z1 = pw4 alors
             m x1 = une base ayant nom = ']1]'
                 et existe un utilisateur ayant nom = ']2]'
                              et mot-de-passe = ']3]' ; ;
               si existe  x1
          alors  m x5 = formal carte-uti dans buf 1
          lire  dans  buf  1
         m z6  = ident de x5
           si z6  = 'uti' alors
        g un utilisateur x6 de x1
          pour x6
             m nom = nom de x5
             m numero = numero de x5
             m mot-de-passe = mot-de-passe de x5
             m acces-macro de droit-utilisateur =
                        acces-macro de x5
             m acces-langage de droit-utilisateur =
                           acces-langage de x5
             m pw-utilisateur de droit-utilisateur =
                         pw-utilisateur de x5
            i x6
           fin
          sinon i 'carte incorrecte ...'  fin
             sinon i (1) 'base*utilisateur*mot-de-passe'
                  i (+2) 'inconnu ...' ecrire fin
            sinon i (1) 'mot de passe incorrect'
                 i (+2) 'requete interdite ...' ecrire fin
      ]fdef   ?
ok  macro  enregistree : creabtch-util
      ]defmac   modif-util  ] de la base  ]  ]  ]  ]
     ]exp  m  z1  =  ']5]'
        si z1 = pw2 ou z1 = pw3 ou z1 = pw4 alors
          m x1 = une base ayant nom = ']2]'
              et existe un utilisateur ayant nom = ']3]'
                           et mot-de-passe = ']4]' ;;
              si existe x1  alors
                m x2 = un utilisateur
                  ayant nom = ']1]' ; de x1
                  si  existe  x2  alors
                      i nom de x2  m nom de x2 = ext
                      i numero de x2  m numero de x2 = ext
                      i mot-de-passe  de x2
                     m mot-de-passe  de x2 = ext
                      i acces-macro de droit-utilisateur de x2
                m acces-macro de droit-utilisateur de x2 = ext
          i acces-langage de droit-utilisateur de x2
            m acces-langage de droit-utilisateur de x2 = ext
              i pw-utilisateur de droit-utilisateur de x2
             m pw-utilisateur de droit-utilisateur de x2 = ext
                 sinon  i  'utilisateur inconnu'  fin
          sinon i (1) 'base*utilisateur*mot-de-passe'
                 i (+2) 'inconnu ...' ecrire fin
            sinon i (1) 'mot de passe incorrect'
                 i (+2) 'requete interdite ...' ecrire fin
      ]fdef  ?
ok  macro  enregistree : modif-util
      ]defmac   suputil  ] de  la  base  ]  ]  ]  ]
     ]exp  m z1  = ']5]'
        si                z1 = pw3 ou z1 = pw4 alors
          m x1 = une base ayant nom = ']2]'
              et existe un utilisateur ayant nom = ']3]'
                           et mot-de-passe = ']4]' ; ;
             si  existe  x1  alors
               m x2 = un utilisateur ayant nom = ']1]' ; de x1
                 si existe  x2  alors  s x2
                   i (1) 'utilisateur' i(+2) ']1]'
                   i (+2) 'supprime de la base' i (+2) ']2]'
```

```
                    ecrire
                sinon  i 'utilisateur inconnu' fin
        sinon i (1) 'base*utilisateur*mot-de-passe'
              i (+2) 'inconnu ...' ecrire fin
          sinon i (1) 'mot de passe incorrect'
              i (+2) 'requete interdite ...' ecrire fin
      |fdef   ?
ok  macro  enregistree : suputil
      |defmac   supbase    |  | |  ]
     ]exp   m zl   =  ']4]'
        si zl = pw3 ou zl = pw4 alors
          m xl = une base ayant nom = ']1]'
            et existe un utilisateur ayant nom = ']2]'
                 et mot-de-passe = ']3]' ; ;
           si  existe  xl  alors  s  xl
              i (1) 'base'  i (+2) ']1]'
              i ( +2 ) 'supprimee' ecrire
         sinon i (1) 'base*utilisateur*mot-de-passe'
              i (+2) 'inconnu ...' ecrire fin
          sinon i (1) 'mot de passe incorrect'
              i (+2) 'requete interdite ...' ecrire fin
      |fdef   ?
ok  macro  enregistree : supbase
      |defmac   fichier  bdb  ]
      ]exp  m zl =  ']1]'  si  zl =   pw4  alors
         i toute base i pwl i pw2 i pw3 i pw4
          sinon i (1) 'mot de passe incorrect'
              i (+2) 'requete interdite ...' ecrire fin
      |fdef   ?
ok  macro  enregistree : fichier
        |defmac   modif-pw  ]
        ]exp  m zl   = ']1]'
        si pas pw4 ou zl = pw4 alors
              i  pwl   m pwl = ext
              i  pw2   m pw2 = ext
              i  pw3   m pw3 = ext
              i  pw4   m pw4 = ext
      i pwl i pw2 i pw3 i pw4
          sinon i (1) 'mot de passe incorrect'
              i (+2) 'requete interdite ...' ecrire fin
      |fdef  ?
ok  macro  enregistree : modif-pw
     non
     non
0055  personal  an = 1010   ph = 0001  date = 17/11/87-321
h.deb = 17h 50m 10s  h.fin = 17h 53m 56s  time = 00004202  code = 000
lgp = 00110  mem = 00039  lo = 000496   in = 00459   out = 00000
.        EOJ
```

INTRETINERA BAZEI DE BAZE

MODIFICAREA PAROLELOR SISTEM

Aceasta operatie se executa,ca in exemplul nostru,obligatoriu,la generarea bazei de baze, si optional ori de cite ori administratorul considera necesara modificarea parolelor sistem (modificind linia 1 cu o secventa MODIF_PW <pw4>).

```
.        JOB PERSONAL,AN:1010,PN:SOC
.        * MODIFICARE PAROLE SISTEM
.        XPROC PROCGEN,MOD
.        MOD &SUPB:DV:ADO&
.        MOD &JURNAL:* &
.        ENDMOD
.        FETCH LN:BIBIMTSO,GN:1,VN:1,FN:SOCBTCH,                *     1
```

```
.          DV:ADO
.       ASSIGN S,DVT:AD,VS:AD1AD1
.       ASSIGN Q,DVT:AD,VS:AD1AD1
.       LABEL S,AM:OFL,FN:'BAZA DE BAZE'
.       LABEL Q,AM:OFL,FN:'MANEVRA SGBD'
.       * ASSIGN Z,DV:MT00
.       * LABEL Z,FN:'JURNAL'
.       ASSIGN T,DV:AD7
.       * APPROCHE FIN PROCEDURE CATALOGUEE
.       EST                          FIN DE LA PROCEDURE CATALOGUEE
.       OPTION CS'CF:50,DS:4096'
.       RUN TIME:999,NL:500000,AD:0,0,KEY:60
socbtch  started
nb cadre fich: 50
%       SOC BN:SOCRATE
%       RUN FN:0
** socrate started **  v1.5    le: 17/11/87  a 17.56.15.76.
            1     modif-pw socrate ?
pw1:
            2     curs
pw2:
            3     socrate
pw3:
            4     absolvit la
pw4:
            5     ase-bucuresti
pw1:curs
pw2:socrate
pw3:absolvit la
pw4:ase-bucuresti
%       LOGOUT
0057  personal   an = 1010    ph = 0001    date = 17/11/87-321
h.deb = 17h 56m 00s   h.fin = 17h 56m 33s   time = 00000604   code = 000
lgp = 00110   mem = 00038   lo = 000056   in = 00034   out = 00000
.       * &lt;SEPARATOR APEL&gt;
.       EOJ
```

GENERAREA BAZEI DE DATE

In acest exemplu baza de date denumita BDDPERS are dimensiunea fizica de 50 cilindri si este divizata logic (comanda % ALLOC) in subspatiile:
FICH: 30 cilindri cu dimensiunea subpagini de 16 cuvinte;
DICO: 10 cilindri cu dimensiunea subpagini de 4 cuvinte;
PROG: 10 cilindri cu dimensiunea subpagini de 256 cuvinte.

Sistemul va calcula, pentru fiecare subspatiu, numarul de pagini in care este decupat. Pentru baza de date se specifica comenzile VOL si UTI . Operatia de formatare este lansata cu comanda % SYSRUN FN:F, pentru toate subspatiile.

```
.       * DEFINIRE CARACTERISTICI TEHNICE PENTRU BAZA DE DATE
.       * ALOCARE SPATIU FIZIC PENTRU BAZA DE DATE
.       ALLOC DVT:AD,VS:AD1AD1,AM:ANY,SZ:50,FN:'BDDPERS'
.       * DEFINIRE UTILIZATORI SI DREPTURI DE ACCES LA BAZA DE DATE
.       * FORMATARE SPATIU FIZIC SI DECUPARE IN SPATII LOGICE
.       XPROC PROCGEN,MOD
.       MOD &JURNAL:* &
.       MOD &SUPB:DV:ADO&
.       MOD &MODUL:SOCGBAS&
.       ENDMOD
.       FETCH LN:BIBIMTSO,GN:1,VN:1,FN:SOCGBAS,                    *
.          DV:ADO
.       ASSIGN S,DVT:AD,VS:AD1AD1
.       ASSIGN Q,DVT:AD,VS:AD1AD1
.       LABEL S,AM:OFL,FN:'BAZA DE BAZE'
.       LABEL Q,AM:OFL,FN:'MANEVRA SGBD'
.       * ASSIGN Z,DV:MT00
.       * LABEL Z,FN:'JURNAL'
```

```
.        ASSIGN T,DV:AD7
.        * APPROCHE FIN PROCEDURE CATALOGUEE
.        EST                          FIN DE LA PROCEDURE CATALOGUEE
.        OPTION CS'CF:50,DS:4096'
.        RUN TIME:999,NL:500000,AD:0,0,KEY:60
socgbas  started
nb cadre fich: 50
%        SOC BN:BDDPERS
         1   vol md50               050
         2   uti ase           0001 ase     imt+comp+mac autorise    00010101
         3   uti avram         0001 soc     imt+comp+mac definition  00000001
         4   eof
%        ALLOC FA:(30,16),DA:10,PA:10

type:fich nbpag: 3593/type:dico nbpag: 1193/type:prog nbpag: 1193 { generation ok.}

%        SYSRUN FN:F

%        LOGOUT
.        EOJ
```

EDITAREA DATELOR TEHNICE ASCIATE BAZELOR DE DATE
SI UTILIZATORILOR GESTIONATI DE BAZA DE BAZE

In acest exemplu sint apelate trei programe de serviciu intr-o singura sesiune de lucru si sub acelasi apel de procesor :

```
.        XPROC PROCGEN,MOD
.        MOD &SUPB:DV:ADO&
.        ENDMOD
.        FETCH LN:BIBIMTSO,GN:1,VN:1,FN:SOCBTCH,DV:ADO
.        ASSIGN S,DVT:AD,VS:AD1AD1
.        ASSIGN Q,DVT:AD,VS:AD1AD1
.        LABEL S,AM:OFL,FN:'BAZA DE BAZE'
.        LABEL Q,AM:OFL,FN:'MANEVRA SGBD'
.        ASSIGN T,DV:AD7
.        * APPROCHE FIN PROCEDURE CATALOGUEE
.        EST                          FIN DE LA PROCEDURE CATALOGUEE
.        OPTION,CS'CF:50,DS:4096'
.        RUN TIME:999,NL:500000,AD:0,0,KEY:60

socbtch  started nb cadre fich: 50

%        SOC BN:SOCRATE
%        RUN FN:0

** socrate started **  v1.5    le: 17/11/87  a 17.58.52.54.

 1      edibase bddpers ase ase ase-bucuresti

 2      creabtch-util de la base bddpers ase ase ase-bucuresti

 3    fichier bdb ase-bucuresti
 4    ?
base  1
 nom:bddpers
  beb
   org-struc: 256  org-fich: 65536  org-mac: 1879048192
     xbeb  1
      type:fich  nbpag:3593 tspag:15 longsp:60 lgchbit:480 dep:64 sl11:4

                 sllpp2:6 srlpp2:1030 sl132mk:18 srl32mk:1042  pp2mask:60

                 kmask:-262144 orgrel:0       nb-cyl: 30
```

```
    xbeb  2
     type:dico  nbpag:1193 tspag:3 longsp:12 lgchbit:160 dep:256 sll1:2

                sllpp2:8 srlpp2:1032 sll32mk:19 srl32mk:1043 pp2mask:252

                kmask:-524288 orgrel:14400   nb-cyl: 10
    xbeb  3
     type:prog  nbpag:1193 tspag:255 longsp:1020 lgchbit:160 dep:4 sll1:8

                sllpp2:2 srlpp2:1026 sll32mk:13 srl32mk:1037 pp2mask:0

                kmask:-8192 orgrel:19200         nb-cyl: 10
support
 type:md50
 volum  1
     nom-proprio:
     nb-cyl: 50

   utilisateur  1
     nom:ase
     numero: 1
     mot-de-passe:ase
     droit-utilisateur
        acces-macro:imt+comp+mac
        acces-langage:autorise
        pw-utilisateur: 10101

   utilisateur  2
     nom:avram
     numero: 1
     mot-de-passe:soc
     droit-utilisateur
        acces-macro:imt+comp+mac
        acces-langage:definition
        pw-utilisateur: 1

   5   uti soc              0001 soc   imt+comp+mac definition  000000

  utilisateur  3
     nom:soc
     numero: 1
     mot-de-passe:soc
     droit-utilisateur
        acces-macro:imt+comp+mac
        acces-langage:definition
        pw-utilisateur: 1

 base  1
  nom:bddpers
  beb
  org-struc: 256        org-fich: 65536        org-mac: 1879048192
   xbeb  1
    type:fich  nbpag:3593 tspag:15 longsp:60 lgchbit:480 dep:64 sll1:4

               sllpp2:6 srlpp2:1030 sll32mk:18 srl32mk:1042 pp2mask:60

               kmask:-262144 orgrel:0      nb-cyl: 30
   xbeb  2
    type:dico  nbpag:1193 tspag:3 longsp:12 lgchbit:160 dep:256 sll1:2

               sllpp2:8 srlpp2:1032 sll32mk:19 srl32mk:1043 pp2mask:252

               kmask:-524288 orgrel:14400  nb-cyl: 10
   xbeb  3
    type:prog  nbpag:1193 tspag:255 longsp:1020 lgchbit:160 dep:4 sll1:8

               sllpp2:2 srlpp2:1026 sll32mk:13 srl32mk:1037 pp2mask:0
```

```
            kmask:-8192 orgrel:19200    nb-cyl: 10
   support
       type:md50
       volume  1
          nom-proprio:
          nb-cyl: 50

 utilisateur  1
       nom:ase
       numero: 1
       mot-de-passe:ase
       droit-utilisateur
          acces-macro:imt+comp+mac

          acces-langage:autorise
          pw-utilisateur: 10101

   utilisateur  2
       nom:avram
       numero: 1
       mot-de-passe:soc
       droit-utilisateur
          acces-macro:imt+comp+mac
          acces-langage:definition
          pw-utilisateur: 1

   utilisateur  3
       nom:soc
       numero: 1
       mot-de-passe:soc
       droit-utilisateur
          acces-macro:imt+comp+mac
          acces-langage:definition
          pw-utilisateur: 1

pw1:curs

pw2:socrate

pw3:absolvit la

pw4:ase-bucuresti

eof                                                                        %%

%          LOGOUT

  0059  personal   an = 1010   ph = 0001   date = 17/11/87-321

  h.deb = 17h 58m 34s   h.fin = 17h 59m 20s   time = 00001307   code = 000

  lgp = 00110   mem = 00038   lo = 000216   in = 00033   out = 00000

          EOJ PERSONAL,AN:1010,PN:,SOC
```

CATALOGAREA PROCEDURILOR SI FORMAREA BIBLIOTECII DE PROCEDURI A APLICATIEI

Catalogarea , intr-o biblioteca sursa , sub forma de proceduri catalogate a structurii conceptuale (STRUCT), a programelor definite (MACRO) si a unei proceduri generale de apel a modulului SOCGBAS (GBASGEN).

```
.       DELETE DV:AD7,FN:'PERSONALSOU'
.       * ALOCARE BIBLIOTECA SURSA PENTRU APLICATIETE
.       ALLOC DV:AD7,FN:'PERSONALSOU',AM:ANY,SZ:5
.       SYSRUN BIBLIO
% OPNLIB R,DV:AD7,GN:1,VN:0,LN:PERSONAL,FT:SOU,INIT
% INCLUDE OL:*1,DL:R,OF:STRUCT
fichier cree      nom = struct  ,numero de mise a jour =  1
% INCLUDE OL:*1,DL:R,OF:GBASGEN
fichier cree      nom = gbasgen ,numero de mise a jour =  1
% INCLUDE OL:*1,DL:R,OF:MACRO
fichier cree      nom = macro   ,numero de mise a jour =  1
% ENDLIB
  0061  personal   an = 1010   ph = 0001   date = 17/11/87-321
  h.deb = 17h 59m 36s   h.fin = 18h 01m 23s   time = 00000752   code = 000
  lgp = 00110   mem = 00016   lo = 000032   in = 00582   out = 00000
.       *
.       SYSRUN DISPLAY
display  started
display         17/11/87  18.01
n o m   d e   l a   b i b l i o t h e q u e   :          personal
          n u m e r o   d e   v e r s i o n   :          00
  n u m e r o   d e   g e n e r a t i o n     :          0001
                              f o r m a t     :          sou
      n o m   d u   p e r i p h e r i q u e   :          ad7
display         17/11/87  18.01
nom du fichier :          numero de mise a jour :          date de creation :
              gbasgen                      001
              macro                        001
              struct                       001

display 17/11/87  18.01 nom du fichier : gbasgen  numero de mise a jour : 001

      .       xproc procgen,mod
      .       mod &modul:socgbas&
      .       mod &supb:dv:ad0&
      .       endmod
      .       option cs'cf:50,ds:4096'
      .       run time:999,nl:500000,ad:0,0,key:60
      &%&       soc bn:&bn:bddpers&,pn:&pn:avram&,an:&an:0001&,pw:&pw:soc&
      &%&       sysrun fn:&functia:s&,st:&st:all&&bs&
      &%&       logout

display 17/11/87  18.01 nom du fichier : macro             numero de mise a jour : 001

      .       xproc procgen,mod
      .       mod &supb:dv:ad0&
      .       endmod
      .       option cs'cf:50,ds:4096'
      .       run time:999,nl:500000,ad:0,0,key:60
      &%&       soc bn:&bn:bddpers&,pn:&pn:avram&,an:&an:0001&,pw:&pw:soc&
      &%&       run fn:m,&lst:lst&
      :defpro pageskip :exp i '-' :fdef?
      &%&       run fn:m,&lst:lst&
      :supexp linast120 ?
      &%&       run fn:m,&lst:lst&
      :defmac linast120 :exp
            i ( 1) '**********************************'
            i (+0) '**********************************'
            i (+0) '**********************************'
```

```
        i (+0) '*********************************'
     ecrire
  :fdef?
  &%&        run fn:m,&lst:lst&
  :supexp linast80
  ?
  &%&        run fn:m,&lst:lst&
  :defmac linast80 : exp
  i (1) '*********************************'
   i(+0) '*********************************'
   i(+0) '*********************'
     ecrire
  :fdef?
  &%&        run fn:m,&lst:lst&
  :supexp  capnom          ?
  &%&        run fn:m,&lst:lst&
  :defmac
   capnom
  :exp
  i ' ' m y2 = 0
   si y1 > 0
    alors
     i (71) 'pagina:' i (80 -4) y1 ecrire
     m y2 = y2 + 1
   fin
   linast80
   i (1) '*  c o d *' i (30) 'd e n u m i r e' i(80) '*'
display  17/11/87  18.01 nom du fichier : macro
   ecrire
    linast80
  m y2 = y2 + 4
  :fdef?
  &%&        run fn:m,&lst:lst&
  :supexp  lisnom          ?
  &%&        run fn:m,&lst:lst&
  :defmac
   lisnom : denumit : cu : rinduri pe pagina
  :exp
  faire
  d x1
   exec pageskip
    i ' ' i (15) 'nomenclatorul de' i (+1) ':2:' ecrire
   capnom
   m y2 = y2 + 2
   m y3 = :3: m y3 = y3 - 1
   pour tout :1: x1
    si y2 >= y3
     alors
      m y1 = y1 + 1
      exec pageskip
      capnom
    fin
    i (1) '*'
    i (9 -7) cod
    i (10) '*'
    i (12) den1
    i (+0) den2
    i (80) '*'
     ecrire
    linast80
    m y2 = y2 + 2
   fin
  :fdef?
  &%&        run fn:m,&lst:lst&
```

```
        :supexp cartela?
        &&&          run fn:m,&lst:lst&
        :defmac cartela :exp macheta :fdef?
        :supexp act-p01 ?
        &&&          run fn:m,&lst:lst&
        &&&          run fn:m,&lst:lst&

display  17/11/87  18.01 nom du fichier : macro          6

        :defpro  act-p01
        :contxt  d x2 = un persoana /* x2 = parametru de apel     */
        :exp  m x4 = u
         m x1 = formal pers dans buf 1
         pour x2  /* citatiile din<stinga> in context<persoana>  */
          m z1 = u z1 = nume de x1
          si existe z1 /* modif,conditionala datorata dublului */
            alors    /* rol al machetei<<creere-actualizare>> */
             m nume = u              /* daca..nume de x1..     */
             m nume = nume de x1     /* este spatiu            */
            fin                      /* z1 ramine la           */
          m z1 = u m z1 = prenume de x1      /* valoarea u     */
          si existe z1               /* ======================*/
            alors
              m prenume = u
              m prenume = prenume de x1'
            fin
          si pas an de data-nasterii
            alors m y1 = u
           m y1 = an-nastere de x1
              m y1 = y1 + 1900
              m an de data-nasterii = y1
              m luna de data-nasterii = luna-nastere de x1
              m zi de data-nasterii = zi-nastere de x1
            fin
            /* incarcare refere <<pers-locn>>  */
            m y1 = u m y1 = localit de x1
            m x4 = u m x4 = un localitati avec cod = y1;
            si existe x4
             alors
              m pers-locn de data-nasterii = u /*evit.incoerenta*/
              m pers-locn de data-nasterii = x4
            fin
            m z1 = u m z1 = cetatenie de x1
            si existe z1
             alors
              m cetatenie = u
              m cetatenie = cetatenie de x1
            fin
            m z1 = u m z1 = nationalitate de x1
            si existe z1
             alors

display 17/11/87  18.01 nom du fichier : macro

              m nationalitate = u
              m nationalitate = nationalitate de x1
            fin
            m z1 = u m z1 = stare-civila de x1
            si existe z1
             alors
              m stare-civila de x2 = u /* contextul<x2> implicit*/
              m stare-civila = stare-civila de x1
            fin
            m z1 = u m z1 = tip-apart de x1
            si existe z1
             alors
              m apartenenta = u
```

```
          m apartenenta = tip-apart de x1
          m an   de data-pcr = an-apart de x1
          m z1   de data-pcr = zz-apart de x1
          m luna de data-pcr = ll-apart de x1
        fin
        m z1 = u m z1 = seria-livret de x1
        si existe z1
          alors
          m seria de livret de situatia-militara =
                         seria-livret de x1
          m nr   de livret de situatia-militara =
                            nr-livret de x1
        fin
      fin /* pentru <pour x2> */
    :fdef
    ?
    &%&          run fn:m,&lst:lst&
    :supexp act-p02 ?
    &%&          run fn:m,&lst:lst&
    :defpro act-p02
    :contxt d x2 = un persoana
    :exp m x1 = formal pers dans buf 1
      pour x2
        i (1  )           ' scrieti programul pentru'
        i (+1 )           'tratarea machetei<p02>;'
        i (+5 )           'eventual modificati programul'
        i (+0 )           'principal pentru a trata'
          ecrire
        i (2  30)         'si macheta<p03>;scrieti secventa'
display 17/11/87  18.01 nom du fichier : macro

        i (+0 )               'de program corespondenta.'
            ecrire     i '?(mult succes)?'
        i (1    ) '<marca>;'  m z2 = matricol de x1 /* inutil */
        i (+15 -13) z2  i (+1 ) '<nume>:' i (+0 16) nume ecrire
      fin
    :fdef ?
    &%&          run fn:m,&lst:lst&
    :lisall ?
    &%&          logout
display 17/11/87  18.01 nom du fichier : struct                  9

    .    xproc procgen,mod
    .    mod &supb:dv:ad0&
    .    endmod
    .    option cs'cf:50,ds:4096'
    .    run time:999,nl:500000,ad:0,0,key:60
    &%&   soc bn:&bn:bddpers&,pn:&pn:avram&,an:&an:0001&,pw:&pw:soc&
    &%&   acti tb:64
    &%&   run fn:d
    debut
    /* variabile de lucru                                          */
    raspuns mot
      rasp  mot 15
    continuati (4 2) (da nu y n )
    mnemonica mot 4
    codnum   de 1 a 999999999
    codalf   mot
    caracter (64 1) (1 2 3 4 5 6 7 8 9 a b c d e f   )
    /*------------------------------------------------------------*/
    /*                                                             */
    muncitori     inverse tout persoana
    tesa          inverse tout persoana
    invgen        inverse tout persoana
    barbati       inverse tout persoana
```

```
       femei        inverse tout persoana
       /*-----------------------------------------------------------*/
       /*                                                           */
       /*  entitatea : <persoana>                                   */
       /*            <<              >>                             */
       /* ------------<< nucleu >>    -----------------------------*/
       entite 36864    persoana
        debut
         /* <cod>::= saalljjnnnc                                    */
         /* -s: sex/secol;                                          */
         /* -aallzz: data nasterii;                                 */
         /* -jj: judetul; -nnn: numar de ordine;                    */
         /* -c: cifra de control.                                   */
         cod         mot 13 avec cle unique fin
         nume        mot 16 avec cle ordonne fin
         prenume     mot 16
         data-nasterii
          debut
           an    de 1900 a 2100

display 17/11/87  18.01 nom du fichier : struct
           luna de 1 a 12
           zi    de 1 a 31
           pers-locn refere locn-pers de un localitati
         fin
         cetatenie        (15 1) (1 2 3 4 5 6 7 8 9 a b c d e f )
         nationalitate    (15 1) (1 2 3 4 5 6 7 8 9 a b c d e f )
         stare-civila     (4  1) (1 2 3 4 )
         apartenenta      (4 5)  (pcr utc odus nm)
         sanct-pol        (15 3) (ep vb vba av mus ms )
         data-pcr
          debut
           an    de 1921 a 2100
           luna de 1 a 12
           zi    de 1 a 31
         fin
         situatia-militara
          debut
           livret
            debut
             seria mot 3
             nr     de  0  a 999999
           fin
           specialitate de 0  a 999
           comisariat   de 0  a 99
           /* comisariat := cod judet ; */
           grad         de 0  a 99
           obligarii    (7  1) ( 1 2 3 4 5 6 7 )
          fin
         grupa-sanguina (4  4) (oi aii biii abiv)
         domiciliu
          debut
           stabil
            debut
             loc-stab refere stab-loc de un localitati
             str      mot 26
             nr       mot 4
             bl       mot 3
             sc       mot 2
             etaj     de 0 a 32 (64)
             ap       de 0 a 999
             mediu    (2 1) (1 2)
             naveta

display 17/11/87  18.01 nom du fichier : struct
```

```
          debut
            distanta de 0 a 99
            an-apr   de 1900  a  2100
            luna-apr de 1 a 12
            ziua-apr de 1 a 31
          fin
        locuinta
          debut
            supr-loc     de 1 a 999
            nr-camere    de 1 a 32
            nr-persoane de 1 a 99
            proprietate (7  1) (1 2 3 4 5 6 7 )
        fin
        buletin-ident
          debut
            seria       mot 2
            /* compilatorul <ldd> accepta si linii vide */
            nr          de 0 a 999999
            an-emitere  de 1950 a 2100
          fin
       fin
       flotant
        debut
          loc-flot refere flot-loc de un localitati
          str        mot 26
          nr         mot 4
          bl         mot 3
          sc         mot 2
          etaj       de 0 a 32 (64)
          ap         de 0 a 99
          mediu      (2  1) (1 2 )
             /* 1: urban; 2: rural. */
        fin
     fin
     /* copii persoanei          */
     pers-copii      anneau
     sot             refere un persoana
     pers-fun        refere fun-pers de un functii
     pers-pro        refere pro-pers de un profesii
     pers-spe        refere spe-pers de un specialitati
     pers-san        refere san-pers de un sanctiuni
     pers-rec        anneau

display 17/11/87  18.01 nom du fichier : struct

               avec chaine double fin
     entite 3 calificative
       debut
        an       de 1977 a 2100
        calific (4 2) (fb b ns st)
                        /* fb: foarte bun; */
                        /* b : bun;        */
                        /* ns: nesatisfacator; */
                        /* st: satisfacator.   */
       fin
     fin
     /*  entitatea : <judete>                        */
     entite (64) judete
      debut
       jud-loc  anneau
       cod      de 1 a 99  avec cle unique fin
       denl     mot
     fin
     /*   entitatea : <localitai>                    */
     entite  16000 localitati
      debut
       loc-jud     refere  jud-loc de un judete
```

```
    /* localitate de care apartine ierarhic */
    sup-loc     refere  loc-sup de un localitati
    /* localitati subordonate administrativ  */
    loc-sup     anneau
    /* persoanele nascute in aceiasi localitate */
    locn-pers   anneau
    /* persoanele cu domiciliul stabil in localitate */
    stab-loc    anneau
    /* persoanele cu domiciliul flotant in localitate */
    flot-loc    anneau
    cod         de 1 a 999999 avec cle unique fin
    den1        mot
    den2        mot ( 10 )
  fin
  /* entitatea : <grade-militare>              */
  entite 128 grade-militare
   debut
    cod   de 0 a 99 avec cle unique fin
    den1  mot
  fin

display 17/11/87  18.01 nom du fichier : struct

  /* entitatea : <nationalitate-cetatenie>                 */
  entite ( 32 )  nationalitate-cetatenie
   debut
    /* cheia realizarii este <numar de realizare>          */
    den1  mot
  fin
  /* entitatea : <functii>                                 */
  entite   4096  functii
   debut
    fun-spe    anneau
    fun-pers   anneau
    cod        de 1 a 9999999 avec cle unique fin
    den1       mot
    den2       mot
  fin
  /* entitatea : <specialitati>                            */
  entite   4096  specialitati
   debut
    spe-fun    refere  fun-spe de un specialitati
    spe-pers   anneau
    cod        de 1 a 9999999 avec cle unique fin
    den1       mot
    den2       mot
  fin
  entite  320000  copii
   debut
    copii-pers          refere oers-copii de un persoana
    nr-copil            de 1 a 32
    nume                mot 16
    prenume             mot 14
    loc-nastere         de 1 a 999999
    data-nasterii
     debut
      an   de 1916 a 2100
      zi   de 1 a 31
      luna de 1 a 32
    fin
    rang-alocatie de 0 a 99
    invaliditate  de 0 a 9
    sistare-aloc  de 0 a 9
    informatii    texte  (4  60)
  fin

display 17/11/87  18.01 nom du fichier : struct
```

```
      /* ---------------------------------------------------------*/
     /*                  <<atentie>>                              */
     /*    urmeaza doua descrieri de relatii ''m <---> n''        */
     /* ---------------------- -----    --------------------------*/
      /* entitatea  : <sanctiune>                                 */
      entite  128 sanctiune
       debut
        sanct anneau
        cod   de 1 a 99 avec cle unique fin
        den1  mot
        den2  mot
      fin
      /* entitatea : <sanctiuni>                                  */
      entite 160000 sanctiuni
       debut
        san-sanct  refere sanct de un sanctiune
        san-pers   refere pers-san de un persoana
        nr-san     de 0 a 99
        procent    de 0 a 99
        cl-cat     de 0 a 99
        gr-tr      de 0 a 99
        retributie de 0 a 29999
        functie    de 1 a 4096
           /* atentie ] este numarul de realizare                 */
           /* ------------------------------------------------   */
        an         de 1944 a 2100
        luna       de 1 a 12
        zi         de 1 a 31
        motiv      texte (255  60)
      fin
      /* entitatea : <profesii>                                   */
      entite  128 profesii
       debut
        pro-per  anneau
        cod      de 0 a 99 avec cle unique fin
        den1     mot
      fin
      /* entitatea : <recompensa>                                 */
      entite  128  recompensa
       debut
        rec-pers anneau avec chaine double fin
        cod      de 1 a 128 avec cle unique fin

display 17/11/87  18.01 nom du fichier : struct

        den1     mot
        den2     mot
      fin
      /* entitatea : <recompense>                                 */
      entite 160000
       debut
        perrec-rec refere rec-pers de un recompensa
                                         avec chaine double fin
        recper-rec refere pers-rec de un persoana
                                         avec chaine double fin
        nr-rec   de 1 a 128
        nr-pers de 1 a 36864
        an       de 1944 a 2100
        luna     de 1 a 12
        zi       de 1 a 31
        motiv    texte (4 60)
       fin
      fin
     ?
     &%&          dacti tb:44
     &%&          dacti tb:64
```

```
&&&        acti tb:49
&&&        run fn:d
d
/*                                                        */
/*  descriere format date de intrare in sistem            */
/* <<<   adaugare la structura                  >>>       */
/*                                                        */
/* -------------------------------------------------------*/
/*   descrierea formatelor pentru incarcare-actualizare   */
/*       entitate : <persoana>                            */
/* -------------------------------------------------------*/
formal pers
 debut
  ident       mot 3
  matricol    mot (13)
  /*-----------------------------------------------------*/
  /*   redefinire componente logice                      */
  /*-----------------------------------------------------*/
      sex             mot      (4  1)
      an-nastere      dilate   (5  2)
      luna-nastere    dilate   (7  2)

display 17/11/87  18.01 nom du fichier : struct
      zi-nastere      dilate   (9  2)
  nume                mot      (16
  prenume             mot       16
  /* < loc nastere >                                     */
    judet             dilate   ( 2 )
    localit           dilate     6
    cetatenie         mot        1
    nationalitate     mot        1
    stare-civila      mot        1
  /*  <apartenenta politica>                             */
    tip-apart         mot        4
    an-apart          dilate     4
    ll-apart          dilate     2
    zz-apart          dilate     2
  /* <livret-militar>                                    */
    seria-livret      mot        3
    nr-livret         dilate   (   6    )
 /* <<<<<< format p02 redefinire p01                     */
  /* situatia militara                                   */
    specialit         dilate   (17  3)
    comisariat        dilate   (20 2 )
    grad              dilate   ( 22 2)
    obligatii         mot       (24  1)
    grupa-sanguina    mot      (25 4 )
  /* <buletin de identitate>                             */
    seria-bi          mot      29 2
    nr-bi             dilate   31 6
    an-emit-bi        dilate   37 4
    den1-circa        mot      (41 20)
    den2-circa        mot      (61 20)
  /*-----------------------------------------------------*/
  /* exercitiu: descrieti un format de intrare pentru    */
  /* restul atributelor asociate entitati <persoana>     */
  /*-----------------------------------------------------*/
   fin
fin
?
&&&        run fn:d
d
/* ------------------------------------------------------*/
/*                                                       */
/*              <<macnom>>                               */
```

```
      /* ''                                   */
      /* ----------      ---------------------*/
      /*                                      */
      /*  adaugare la structura formate       */
      /*       <<nomenclatoare>>              */
      /* -------------------------------------*/
     formal           macheta
      debut
       ident          mot 3
       filler         mot 77
       /* <nationalitate-cetatenie>           */
       cod-nac        mot (4 1)
       denl-nac       mot (5 25)
       /* <grade-militare>                    */
       cod-grm        dilate (4  2)
       denl-grm       mot    (6 27)
       /*        <judete>                     */
       cod-jud        dilate (4  2)
       cifk           dilate (6  1)
       denl-jud       mot    (7 15)
       /*       <localitati>                  */
       refl-jud       dilate (4  2)
       cod-loc        dilate (6  6)
       ref2-loc       dilate (12 6)
       denl-loc       mot    (18 30)
       den2-loc       mot    (48 10)
       /*        <profesii>                   */
       cod-pro        dilate (4  2)
       denl-pro       mot    (6 30)
       den2-pro       mot    (36 30)
       /*      mot<specialitati>              */
       refl-fun       dilate  (4  4)
       cod-spe        dilate  (8  4)
       denl-spe       mot        (12 30)
       den2-spe       mot        (42 30)
       /*        <sanctiuni>                  */
       cod-san        dilate  (4  3)
       denl-san       mot     (7  30)
       den2-san       mot     (37 30)
       /*          <functii                   */
       cod-fun        dilate  (4  4)
       denl-fun       mot     (8 30)

display 17/11/87  18.01 nom du fichier : struct

       den2-fun       mot     (38 20)
       fin
       /* sfirsit <bloc formal> <macheta>     */
      fin
      /* sfirsit <bloc> definire / adaugare <d>   */
      ?
      &&&     logout
 0061  personal   an = 1010   ph = 0002   date = 17/11/87-321
 h.deb = 18h 01m 23s   h.fin = 18h 01m 45s   time = 00000298   code = 000
 lgp = 00110   mem = 00009   lo = 000616   in = 00003   out = 00000
     EOJ
```

GENERAREA AUTOMATA A PROGRAMELOR DE CREERE SI INTRETINERE A REALIZARILOR ENTITATILOR DE TIP NOMENCLATOR DIN BAZA DE DATE

Este apelat un prototip din generatia 1 pentru entitatile furnizate ca parametru la apel. Rezultatul generarii este procedura CAMIDENT stocata in biblioteca sistem Z%PROC.

```
.        * APEL PROCESOR DE GENERARE PROGRAME
.        XPROC CAM,MOD
.        MOD 3,3
         mpage    <parametrii>
         mstring enti='judete','1','localitati','4','localitati',      *
                 'specialitati','4','functii','profesii','1',          *
                 'functii','3','sanctiune','3','recompensa','3',       *
                 'grade-militare','1'
         mstring tipref='num'
.         ENOMOD

.         COMPILE MAGIRIS.DV:ADO,GN:1,VN:0,                            *    1
.                 LN:Z%PROC,FN:CAMIDENT,NLG,OBL                             2
magiris  started

liste  source        magiris 0.0      17/11/87   18.02             1 \
texte objet catalogue dans la bibliotheque source          z%proc    \vn:0,gn:1
     nouveau fichier cree         camident
           numero de mise a jour        un: 001

liste  objet         magiris 0.0      17/11/87   18.02             4
       %         soc          bn:coral,pw:cc,pn:avram,an:1 <sterge>
       %          run fn:m
       :supexp  cam-jud  ?
       %          run fn:m
        :defmac
          cam-jud
        :exp
         d  x1  d  x2  d  x3  d x4    m y6 = 0   m y7 = 0
         m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
         m x1 = formal cartela  dans  buf  0
         faire
          lire dans buf 0
          m z1 = ident       de          x1
          si z1 = '[illegible]'        alors  sortie  fin
          m  y4 = y4 + 1
          si z1 ^=  'jud'
            alors
             m y7 = y7 + 1
             i '?? ident ?? *eronat* '
             refaire
          fin
          m z5 = u   m y1 = u
          m y1 = cod-jud   de   x1
          m x2 = un  judete  avec cod = y1 ;
         si existe x2
            alors
             m z2 = denl-jud                       de x1
             si z2 = '  '
              alors
               m denl      de  x2    = u
               m cod       de  x2    = u
               s x2
               m y2 = y2 + 1   /* stergeri  */
             sinon
               m denl      de  x2    = denl-jud    de  x1
               m y6 = y6 + 1   /* modificari(actualizari) */
             fin
          sinon
```

```
      g  un  judete         x3
      m  cod           de  x3   = cod-jud       de  x1
      m denl           de  x3  = denl-jud     de x1
      m  y3 = y3 + 1   /*  adaugari(creeri)  */
     fin
   . refaire
     fin
     i (  1   ) ' ** au fost citite : '
liste objet        magiris 0.0      17/11/87   18.02          5
     i (+10 -10)  y4
     i (+1      )      ' inregistrari < jud      > '
     i (+0      )      ' din care: '
     ecrire
     i (+20 -10)   y3
     i (+1      )      ' adaugari(creeri); '
     ecrire
     i (+20 -10)   y2
     i (+1      )      ' stergeri(suprimari); '
     ecrire
     i (+20 -10) y6
     i (+1      )      ' modificari(actualizari); '
     ecrire
     i (+20 -10) y7
     i (+1      )      ' eronate '
     ecrire
   :fdef ?
   %        run fn:m
   :supexp cam-loc ?
   %        run fn:m
   :defmac
     cam-loc
   :exp
     d  x1   d   x2   d  x3   d x4   m y6 = 0    m y7 = 0
     m y1 = 0    m y2 = 0   m y3 = 0   m y4 = 0   m y5 = 0
     m x1 = formal cartela dans  buf 0
     faire
      lire dans buf 0
      m z1 = ident        de            x1
      si z1 = 'xxxx'            alors  sortie  fin
      m   y4 = y4 + 1
      si z1 ^=   'loc'
       alors
       m y7 = y7 + 1
        i '?? ident ?? *eronat* '
       refaire
     fin
     m z5 = u    m y1 = u
     m y1 = cod-loc    de     x1
     m x2 = un   localitati   avec cod = y1 ;
   si existe x2
     alors
     m z2 = denl-loc                           de  x1
     si z2 = '   '
      alors
       m denl          de   x2      = u
liste objet        magiris 0.0      17/11/87   18.02          6
       m cod           de   x2      = u
       m den2          de   x2      = u
       m y2 = y2 + 1    /* stergeri */
      sinon
       m denl          de   x2      = denl-loc          de  x1
       m den2          de   x2      = den2-loc          de  x1
       m y6 = y6 + 1  /* modificari(actualizari) */
      fin
    sinon
     g  un  localitati            x3
```

```
  m  cod          de  x3     = cod-loc       de  x1
  m  den1         de  x3     = den1-loc      de  x1
  m  den2         de  x3     = den2-loc      de  x1
  m y3 = y3 + 1        /*  adaugari(creeri)   */
 fin
  m x2 = un localitati        avec  cod = y1 ;
 m y5 = refl-loc  de x1
  m x4 = un  localitati       avec cod  =  y5  ;
  m loc-loc  de  x2  =  x4
 refaire
 fin
 i (  1   ) ' ** au fost citite : '
 i (+10 -10)  y4
 i (+1     )      ' inregistrari < loc   >'
 i (+0     )      ' din care: '
 ecrire
 i (+20 -10)  y3
 i (+1    )       ' adaugari(creeri); '
 ecrire
 i (+20 -10)  y2
 i (+1    )       ' stergeri(suprimari); '
 ecrire
 i (+20 -10) y6
 i (+1    )       ' modificari(actualizari); '
 ecrire
 i (+20 -10) y7
 i (+1    )       ' eronate  '
 ecrire
:fdef  ?
%     run fn:m
:supexp  cam-spe  ?
%     run fn:m
 :defmac
  cam-spe
 :exp
 d  x1  d  x2  d  x3  d x4    m y6 = 0    m y7 = 0
liste  objet      magiris 0.0     17/11/87  18.02              7
  m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
  m x1 = formal cartela dans  buf 0
  faire
   lire dans buf 0
   m z1 = ident       de           x1
   si z1 = 'xxxx'          alors  sortie  fin
   m  y4 = y4 + 1
   si z1 ^=  'spe'
     alors
      m y7 = y7 + 1
      i '?? ident ?? *eronat*
      refaire
   fin
   m z5 = u  m y1 = u
   m y1 = cod-spe   de    x1
   m x2 = un  specialitati  avec cod = y1 ;
  si exista x2
     alors
     m z2 = den1-spe                      de  x1
     si z2 = '  '
       alors
        m den1       de  x2   = u
        m cod        de  x2   = u
        m den2       de  x2   = u
        m y2 = y2 + 1    /* stergeri */
       sinon
        m den1       de  x2    = den1-spe     de  x1
        m den2       de  x2    = den2-spe     de  x1
        m y6 = y6 + 1   /* modificari(actualizari) */
     fin
```

```
       sinon
        g  un  specialitati       x3
        m  cod            de  x3    = cod-spe        de  x1
        m  den1           de  x3    = den1-spe       de  x1
        m  den2           de  x3    = den2-spe       de  x1
        m y3 = y3 + 1         /*  adaugari(creeri)  */
       fin
      m x2 = un specialitati            avec  cod = y1 ;
     m y5 = refl-spe  de x1
      m x4 = un  functii          avec  cod   = y5 ;
      m spe-fun  de  x2   =   x4
     refaire
     fin
     i (  1   )  ' ** au fost citite : '
     i (+10 -10)  y4
     i (+1    )        ' inregistrari < spe      > '
liste  objet       magiris 0.0      17/11/87    18.02
     i (+0    )        ' din care: '
      ecrire
      i (+20 -10)   y3
      i (+1    )        ' adaugari(creeri); '
      ecrire
      i (+20 -10)   y2
      i (+1    )        ' stergeri(suprimari); '
      ecrire
      i (+20 -10) y6
      i (+1    )        ' modificari(actualizari); '
      ecrire
      i (+20 -10) y7
      i (+1    )        ' eronate  '
      ecrire
     :fdef  ?
    %           run fn:m
    :supexp   cam-pro   ?
    %           run fn:m
     :defmac
      cam-pro
     :exp
      d  x1   d   x2  d   x3   d x4      m  y6 = 0    m  y7 = 0
      m y1 = 0    m y2 = 0   m y3 = 0   m y4 = 0   m y5 = 0
      m x1 = formal cartela  dans  buf  0
      faire
       lire dans buf 0
       m z1 = ident        de             x1
       si z1 = '¤¤¤¤'            alors  sortie  fin
       m  y4 = y4 + 1
       si z1 ^=  'pro'
         alors
          m y7 = y7 + 1
          i '?? ident ?? *eronat* '
          refaire
       fin
       m z5 = u    m y1 = u
       m y1 = cod-pro   de    x1
       m x2 = un  profesii  avec cod = y1 ;
      si exista x2
         alors
          m z2 = den1-pro                    de x1
          si z2 = '   '
            alors
              m den1      de   x2    = u
              m cod       de   x2    = u
              s x2
liste  objet       magiris 0.0      17/11/87    18.02        9
              m y2 = y2 + 1    /* stergeri  */
            sinon
              m den1      de  x2    = den1-pro      de  x1
```

```
          m y6 = y6 + 1   /* modificari(actualizari) */
        fin
     sinon
      g un  profesii         x3
      m  cod           de  x3   = cod-pro   de  x1
      m den1           de  x3   = den1-pro   de x1
      m  y3 = y3 + 1  /*  adaugari(creeri)  */
    fin
   refaire
   fin
   i ( 1  ) ' ** au fost citite : '
   i (+10 -10)  y4
   i (+1     )       ' inregistrari < pro        > '
   i (+0     )       ' din care: '
   ecrire
   i (+20 -10)  y3
   i (+1     )       ' adaugari(creeri); '
   ecrire
   i (+20 -10)  y2
   i (+1     )       ' stergeri(suprimari); '
   ecrire
   i (+20 -10) y6
   i (+1     )       ' modificari(actualizari); '
   ecrire
   i (+20 -10) y7
   i (+1     )       ' eronate  '
   ecrire
  :fdef ?
 %         run fn:m
 :supexp  cam-fun  ?
 %         run fn:m
  :defmac
    cam-fun
  :exp
   d  x1  d  x2  d  x3  d x4       m y6 = 0   m y7 = 0
   m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
   m x1 = formal cartela  dans  buf  0
   faire
    lire dans buf 0
    m z1 = ident      de         x1
    si z1 = '¤¤¤¤'         alors  sortie  fin
    m  y4 = y4 + 1
    si z1 ^= 'fun'
liste  objet       magiris 0.0      17/11/87   18.02
      alors
      m y7 = y7 + 1
      i '?? ident ?? *eronat* '
      refaire
    fin
    m z5 = u   m y1 = u
    m y1 = cod-fun    de    x1
    m x2 = un  functii  avec cod = y1 ;
   si exist x2
      alors
      m z2 = den1-fun              de  x1
      si z2 = '  '
        alors
         m den1        de  x2    = u
         m cod         de  x2    = u
         m den2        de  x2    = u
         m y2 = y2 + 1   /* stergeri */
        sinon
         m den1        de  x2    = den1-fun     de  x1
         m den2        de  x2     = den2-fun          de  x1
         m y6 = y6 + 1   /* modificari(actualizari) */
      fin
    sinon
```

```
        g  un  functii          x3
         m  cod          de  x3    = cod-fun          de  x1
         m  den1         de  x3    = den1-fun         de  x1
         m  den2         de  x3    = den2-fun         de  x1
         m y3 = y3 + 1        /*  adaugari(creeri)  */
        fin
      refaire
      fin
      i (  1  )  ' ** au fost citite : '
      i (+10 -10)  y4
      i (+1      )          ' inregistrari < fun      >'
      i (+0      )          ' din care: '
       ecrire
      i (+20 -10)  y3
      i (+1      )          ' adaugari(creeri); '
       ecrire
      i (+20 -10)  y2
      i (+1      )          ' stergeri(suprimari); '
       ecrire
      i (+20 -10) y6
      i (+1      )          ' modificari(actualizari); '
       ecrire
      i (+20 -10) y7
liste  objet          magiris 0.0      17/11/87   18.02             11
      i (+1      )          ' eronate '
       ecrire
     :fdef  ?
%      run fn:m
:supexp  cam-san  ?
%      run fn:m
 :defmac
  cam-san
:exp
 d  x1   d  x2   d  x3   d x4      m y6 = 0    m y7 = 0
 m y1 = 0    m y2 = 0   m y3 = 0   m y4 = 0  m y5 = 0
 m x1 = formal cartela  dans  buf  0
 faire
  lire dans buf 0
  m z1 = ident      de      x1
  si z1 = '****'          alors  sortie  fin
  m  y4 = y4 + 1
  si z1 ^=  'san'
   alors
    m y7 = y7 + 1
    i '?? ident ?? *eronat* '
    refaire
  fin
  m z5 = u    m y1 = u
  m y1 = cod-san   de    x1
  m x2 = un  sanctiune  avec cod = y1 ;
 si exista x2
  alors
  m z2 = den1-san                          de  x1
  si z2 = ' '
   alors
    m den1       de  x2    = u
    m cod        de  x2    = u
    m den2       de  x2    = u
    m y2 = y2 + 1     /* stergeri */
   sinon
    m den1       de  x2    = den1-san       de  x1
    m den2       de  x2    = den2-san       de  x1
    m y6 = y6 + 1     /* modificari(actualizari) */
  fin
  sinon
   g  un  sanctiune          x3
   m  cod       de  x3    = cod-san        de  x1
```

```
       m  den1          de  x3     = den1-san       de  x1
       m  den2          de  x3     = den2-san       de  x1
       m y3 = y3 + 1        /*  adaugari(creeri)   */
liste  objet        magiris 0.0      17/11/87   18.02             12
      fin
    refaire
    fin
    i (  1  ) ' ** au fost citite : '
    i (+10 -10)  y4
    i (+1      )        ' inregistrari < san     > '
    i (+0      )        ' din care: '
     ecrire
    i (+20 -10)  y3
    i (+1      )        ' adaugari(creeri); '
     ecrire
    i (+20 -10)  y2
    i (+1      )        ' stergeri(suprimari); '
     ecrire
    i (+20 -10)  y6
    i (+1      )        ' modificari(actualizari); '
     ecrire
    i (+20 -10)  y7
    i (+1      )        ' eronate  '
     ecrire
   :fdef  ?
  %        run fn:m
   :supexp  cam-rec  ?
  %        run fn:m
   :defmac
     cam-rec
   :exp
    d  x1  d  x2  d   x3   d x4     m y6 = 0   m y7 = 0
    m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
    m x1 = formal cartela  dans  buf  0
    faire
     lire dans buf 0
     m z1 = ident         de          x1
     si z1 = '¤¤¤¤'          alors  sortie  fin
     m   y4 = y4 + 1
     si z1 ^=  'rec'
        alors
       m y7 = y7 + 1
         i '?? ident ?? *eronat* '
        refaire
     fin
     m z5 = u   m y1 = u
     m y1 = cod-rec    de    x1
     m x2 = un  recompensa  avec cod = y1 ;
    si existe x2
        alors
liste  objet        magiris 0.0      17/11/87   18.02             13
        m z2 = den1-rec                    de  x1
          si z2 = '  '
            alors
             m den1          de  x2    = u
             m cod           de  x2    = u
             m den2          de  x2    = u
             m y2 = y2 + 1   /* stergeri */
            sinon
             m den1          de  x2     = den1-rec          de   x1
             m den2          de  x2     = den2-rec          de   x1
             m y6 = y6 + 1   /* modificari(actualizari) */
         fin
     sinon
      g  un  recompensa          x3
      m   cod            de  x3     = cod-rec           de   x1
      m   den1           de  x3     = den1-rec          de   x1
```

```
     m  den2          de  x3   = den2-rec         de  x1
     m y3 = y3 + 1     /*  adaugari(creeri)  */
    fin
   refaire
   fin
   i( 1  ) ' ** au fost citite : '
   i (+10 -10)  y4
   i (+1    )      ' inregistrari < rec     > '
   i (+0    )      ' din care:'
   ecrire
   i (+20 -10)  y3
   i (+1    )      ' adaugari(creeri); '
   ecrire
   i (+20 -10)  y2
   i (+1    )      ' stergeri(suprimari); '
   ecrire
   i (+20 -10) y6
   i (+1    )      ' modificari(actualizari); '
   ecrire
   i (+20 -10) y7
   i (+1    )      ' eronate '
   ecrire
  :fdef ?
 %          run fn:m
 :supexp  cam-grm  ?
 %          run fn:m
  :defmac
   cam-grm
  :exp
   d  x1  d  x2  d  x3  d x4     m y6 = 0   m y7 = 0
liste  objet        magiris 0.0       17/11/87   18.02            14
   m y1 =0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
   m x1 = formal cartela  dans  buf  0
   faire
    lire dans buf 0
    m z1 = ident        de          x1
    si z1 = '¤¤¤¤'          alors  sortie  fin
    m  y4 = y4 + 1
    si z1 ^=  'grm'
     alors
      m y7 = y7 + 1
      i '?? ident ?? *eronat* '
      refaire
    fin
    m z5 = u   m y1 = u
    m y1 = cod-grm    de    x1
    m x2 = un  grade-militare  avec cod = y1 ;
   si exista x2
    alors
     m z2 = den1-grm                 de x1
     si z2 = '  '
      alors
       m den1       de  x2    = u
       m cod        de  x2    = u
       s x2
       m y2 = y2 + 1   /* stergeri  */
      sinon
       m den1      de  x2   = den1-grm     de  x1
       m y6 = y6 + 1   /* modificari(actualizari) */
     fin
   sinon
    g  un  grade-militare           x3
    m   cod               de  x3    = cod-grm       de  x1
    m den1                de  x3    = den1-grm   de x1
    m   y3 = y3 + 1  /*  adaugari(creeri)   */
   fin
  refaire
```

```
     fin
     i (  1  ) ' ** au fost citite : '
     i (+10 -10)  y4
     i (+1    )   ' inregistrari < grm
     i (+0    )     ' din care: '
      ecrire
     i (+20 -10)  y3
     i (+1    )     ' adaugari(creeri); '
      ecrire
     i (+20 -10)  y2
liste  objet     magiris 0.0       17/11/87   18.02              15
     i (+1    )   ' stergeri(suprimari); '
      ecrire
     i (+20 -10) y6
     i (+1    )   ' modificari(actualizari); '
      ecrire
     i (+20 -10) y7
     i (+1    )     ' eronate  '
      ecrire
   :fdef  ?
  %       logout
  .       ** o!k!<macro>

               *****fin magiris

0062  personal   an = 1010   ph = 0001   date = 17/11/87-321

h.deb = 18h 01m 47s   h.fin = 18h 02m 42s   time = 00001196   code = 002

lgp = 00110   mem = 00018   lo = 000592   in = 00034   out = 00000

.        * APPROCHE FIN PROCEDURE CATALOGUEE

.        EST                              FIN DE LA PROCEDURE CATALOGUEE

.        EOJ PERSONAL,AN:1010,PN:,SOC
```

COMPILAREA STRUCTURII CONCEPTUALE A BAZEI DE DATE

In acest exemplu structura se gaseste, sub forma de procedura de comenzi, in biblioteca de aplicatii denumita PERSONAL, unde a fost catalogata anterior cu ajutorul programului bibliotecar. Pentru aceasta structura se ofera, partial, imaginea memoriei virtuale rezultate (dictionarul de transformare INTERN/EXTERN).

```
.       JOB PERSONAL,AN:1010,PN:SOC
.       * COMPILARE <STRUCTURA CONCEPTUALA> DESCRISA CU LDD SOCRATE
.       * SI OBTINERE DICTIONAR DE TRANSFORMARE INTERN/EXTERN

.       XPROC *,DVT:AD,VS:AD1AD1,VN:0,FN:'PERSONALSOU/STRUCT',MOD
.       MOD 6
%       ACTI TB:49
%       ACTI TB:44
.       MOD 125,125
  pers-san anneau
.       MOD 187,187
    spe-fun refere fun-spe de un functii
.       MOD 195,195
   copii-pers refere pers-copii de un persoana
.       MOD 244,244
  pro-pers  anneau
.       MOD 257,257
 entite  160000 recompense
.       MOD 360,360
  refl-loc   dilate (12  6)
.       ENDMOD

.       XPROC PROCGEN,MOD
.       MOD &SUPB:DV:ADO&
.       ENDMOD

.       FETCH LN:BIBIMTSO,GN:1,vN.1,FN:SOCBTCH,
.               DV:ADO
.       ASSIGN S,DVT:AD,VS:AD1AD1
.       ASSIGN Q,DVT:AD,VS:AD1AD1
.       LABEL S,AM:OFL,FN:'BAZA DE BAZE'
.       LABEL Q,AM:OFL,FN:'MANEVRA SGBD'
.       * ASSIGN Z,DV:MTOO
.       * LABEL Z,FN:'JURNAL'
.       ASSIGN T,DV:AD7
.       * APPROCHE FIN PROCEDURE CATALOGUEE
.       EST                              FIN DE LA PROCEDURE CATALOGUEE

.       OPTION CS'CF:50,DS:4096'
.       RUN TIME:999,NL:500000,AD:0,0,KEY:60

socbtch  started

nb cadre fich: 50

%       SOC BN:BDDPERS,PN:AVRAM,AN:0001,PW:SOC

%       ACTI TB:49

%       ACTI TB:44

%       ACTI TB:64

%       RUN FN:D

** socrate started **  v1.5   le: 17/11/87  a 18.04.31.20.

1   debut
```

```
raspuns mot
  rasp  mot 15
continuati (4 2) (da nu y n )
mnemonica mot 4
codnum   de 1 a 999999999
codalf     mot
caracter (64 1) (1 2 3 4 5 6 7 8 9 a b c d e f  )
/*----------------------------------------------------------*/
/*                                                          */
muncitori      inverse tout persoana
tesa           inverse tout persoana
invgen         inverse tout persoana
barbati        inverse tout persoana
femei          inverse tout persoana
/*----------------------------------------------------------*/
/*                                                          */
/*  entitatea : <persoana>                                  */
/*           <<                  >>                         */
/* -----------<< nucleu >>  --------------------------------*/
entite 36864   persoana
 debut
  /* <cod>::= saalljjnnnc                                   */
  /* -s: sex/secol;                                         */
  /* -aallzz: data nasterii;                                */
  /* -jj: judetul; -nnn: numar de ordine;                   */
  /* -c: cifra de control.                                  */
  cod          mot 13 avec cle unique fin
  nume         mot 16 avec cle ordonne fin
  prenume      mot 16
  data-nasterii
   debut
    an   de 1900 a 2100
    luna de 1 a 12
    zi   de 1 a 31
    pers-locn refere locn-pers de un localitati
  fin
  cetatenie       (15 1) (1 2 3 4 5 6 7 8 9 a b c d e f )
  nationalitate   (15 1) (1 2 3 4 5 6 7 8 9 a b c d e f )
  stare-civila    (4  1) (1 2 3 4 )
  apartenenta     (4 5)  (pcr utc odus nm)
  sanct-pol       (15 3) (ep vb vba av mus ms )
  data-pcr
   debut
    an   de 1921 a 2100
    luna de 1 a 12
    zi   de 1 a 31
  fin
  situatia-militara
   debut
    livret
     debut
      seria mot 3
      nr    de  0  a 999999
    fin
    specialitate  de 0  a 999
    comisariat    de 0  a 99
    /* comisariat := cod judet ; */
    grad          de 0  a 99
    obligarii   (7  1) ( 1 2 3 4 5 6 7 )
   fin
   grupa-sanguina (4  4) (oi aii biii abiv)
   domiciliu
    debut
     stabil
      debut
       loc-stab refere stab-loc de un localitati
```

```
        str         mot 26
        nr          mot 4
        bl          mot 3
        sc          mot 2
        etaj        de  0 a 32 (64)
        ap          de  0 a 999
        mediu       (2  1) (1 2)
        naveta
         debut
          distanta de 0 a 99
          an-apr   de 1900  a  2100
          luna-apr de 1 a 12
          ziua-apr de 1 a 31
        fin
        locuinta
         debut
          supr-loc     de 1 a 999
          nr-camere    de 1 a 32
          nr-persoane  de 1 a 99
          proprietate  (7  1) (1 2 3 4 5 6 7 )
        fin
        buletin-ident
         debut
          seria        mot 2
          /* compilatorul <ldd> accepta si linii vide */
          nr           de 0 a 999999
          an-emitere   de 1950 a 2100
         fin
       fin
       flotant
        debut
         loc-flot refere flot-loc de un localitati
         str       mot 26
         nr        mot 4
         bl        mot 3
         sc        mot 2
         etaj      de 0 a 32 (64)
         ap        de 0 a 99
         mediu     (2  1) (1 2 )
              /* 1: urban; 2: rural. */
       fin
      fin
      /* copii persoanei                */
      pers-copii    anneau
      sot           refere un persoana
      pers-fun      refere fun-pers de un functii
      pers-pro      refere pro-pers de un profesii
      pers-spe      refere spe-pers de un specialitati
     pers-san anneau
      pers-rec      anneau
           avec chaine double fin
      entite 3 calificative
       debut
        an     de 1977 a 2100
        calific (4  2) (fb b ns st)
                         /* fb: foarte bun; */
                         /* b : bun;        */
                         /* ns: nesatisfacator; */
                         /* st: satisfacator.   */
      fin
     fin
    /*   entitatea : <judete>              */
    entite    (64)      judete
     debut
      jud-loc   anneau
      cod       de 1 a 99  avec cle unique fin
      den1      mot
```

```
      fin
      /*   entitatea : <localitai>                    */
      entite  16000 localitati
       debut
         loc-jud     refere  jud-loc de un judete
         /* localitate de care apartine ierarhic */
         sup-loc     refere  loc-sup de un localitati
         /* localitati subordonate administrativ   */
         loc-sup     anneau
         /* persoanele nascute in aceiasi localitate */
         locn-pers   anneau
         /* persoanele cu domiciliul stabil in localitate */
         stab-loc    anneau
         /* persoanele cu domiciliul flotant in localitate */
         flot-loc    anneau
         cod         de 1 a 999999 avec cle unique fin
         den1        mot
         den2        mot ( 10 )
      fin
      /* entitatea : <grade-militare>               */
      entite 128 grade-militare
       debut
         cod   de 0 a 99 avec cle unique fin
         den1  mot
      fin
      /* entitatea : <nationalitate-cetatenie>               */
      entite ( 32 )  nationalitate-cetatenie
       debut
         /* cheia realizarii este <numar de realizare>  */
         den1   mot
      fin
      /* entitatea : <functii>                                */
      entite   4096  functii
       debut
         fun-spe   anneau
         fun-pers  anneau
         cod       de 1 a 9999999 avec cle unique fin
         den1      mot
         den2      mot
      fin
      /* entitatea : <specialitati>                           */
      entite   4096  specialitati
       debut
         spe-fun refere fun-spe de un functii
         spe-pers  anneau
         cod       de 1 a 9999999 avec cle unique fin
         den1      mot
         den2      mot
      fin
      entite  320000   copii
       debut
         copii-pers refere pers-copii de un persoana
         nr-copil         de 1 a 32
         nume              mot 16
         prenume           mot 14
         loc-nastere      de 1 a 999999
         data-nasterii
          debut
           an    de 1916 a 2100
           zi    de 1 a 31
           luna de 1 a 32
        fin
         rang-alocatie de 0 a 99
         invaliditate  de 0 a 9
         sistare-aloc  de 0 a 9
         informatii    texte  (4  60)
       fin
```

```
/* ---------------------------------------------------------*/
/*                    <<atentie>>                           */
/*    urmeaza doua descrieri de relatii ''m <--> n''        */
/* ---------------------------------------------------------*/
/* entitatea   : <sanctiune>                                */
entite  128 sanctiune
 debut
  sanct anneau
  cod   de 1 a 99 avec cle unique fin
  den1  mot
  den2  mot
fin
/* entitatea : <sanctiuni>                                  */
entite 160000 sanctiuni
 debut
  san-sanct  refere sanct de un sanctiune
  san-pers   refere pers-san de un persoana
  nr-san     de 0 a 99
  procent    de 0 a 99
  cl-cat     de 0 a 99
  gr-tr      de 0 a 99
  retributie de 0 a 29999
  functie    de 1 a 4096
     /* atentie ] este numarul de realizare              */
     /* ------------------------------------             */
  an         de 1944 a 2100
  luna       de 1 a 12
  zi         de 1 a 31
  motiv      texte (255  60)
fin
/* entitatea : <profesii>                                   */
entite  128 profesii
 debut
 pro-pers  anneau
  cod      de 0 a 99 avec cle unique fin
  den1     mot
fin
/* entitatea : <recompensa>                                 */
entite  128  recompensa
 debut
  rec-pers anneau avec chaine double fin
  cod      de 1 a 128 avec cle unique fin
  den1     mot
  den2     mot
fin
/* entitatea : <recompense>                                 */
entite  160000 recompense

 debut
  perrec-rec refere rec-pers de un recompensa
                                avec chaine double fin
  recper-rec refere pers-rec de un persoana
                                avec chaine double fin
  nr-rec   de 1 a 128
  nr-pers  de 1 a 36864
  an       de 1944 a 2100
  luna     de 1 a 12
  zi       de 1 a 31
  motiv    texte (4  60)
 fin
fin
?
```

CODIFICAREA INTERNĂ A STRUCTURII (IMAGINE PARȚIALĂ)

adresse	identificateur	frer-per	fils	typ	cmax	origine	tail ptpr	org adrdic	nbmoy
00000100	fichier*********	00000000	0000013c	19	1	65536	26cd9349**	00000000**	00000000**
0000013c	raspuns*********	00000178	00000000	15	0	65536	001f8000**	00000000**	00000000**
00000178	rasp************	000001b4	00000000	15	0	65543	18108000**	00000000**	00000000**
000001b4	continuati******	0000022c	000001f0	16	4	65547	18038000**	00000000**	00040004**
000001f0 Bda Bnu Ay An									
0000022c	mnemonica*******	00000268	00000000	15	0	65548	00058000**	00000000**	00000000**
00000268	codnum**********	000002a4	00000001	10	0	65550	001e8000**	00000000**	3b9ac9fe**
000002a4	codalf**********	000002e0	00000000	15	0	65551	001f8000**	00000000**	00000000**
000002e0	caracter********	00000448	0000031c	16	64	65558	18078000**	00000000**	0004000f**
0000031c 1 2 3 4 5 6 7 8 9 a b c d e f									
00000358									
00000394									
000003d0									
0000040c									
00000448	muncitori*******	000004c0	000006a0	3*	36864	65559	00000001**	10000481**	00000000**
00000484									
persoanai 7 8 9 a b c d e f									
000004c0	tesa************	00000538	000006a0	3*	36864	66713	00000001**	10000481**	00000000**
000004fc d									
persoanai 7 8 9 a b c d e f									
00000538	invgen**********	000005b0	000006a0	3*	36864	67867	00000001**	10000481**	00000000**
00000574									
persoanai 7 8 9 a b c d e f									
000005b0	barbati*********	00000628	000006a0	3*	36864	69021	00000001**	10000481**	00000000**
000005ec									
persoanai 7 8 9 a b c d e f									
00000628	femei***********	000006a0	000006a0	3*	36864	70175	00000001**	10000481**	00000000**
00000664									
persoanai 7 8 9 a b c d e f									
000006a0	persoana********	00001bb8	000006dc	2*	36864	73635	00000043**	10000481**	00000000**
000006dc	persoana********	fffff960	00000718	19	1	0	00000043**	00000000**	00000000**
00000718	cod*************	00000754	00000000	15	73727	0	000e8000**	9000004e**	00000000**
00000754	nume************	00000790	00000000	15	73727	3	10118000**	840000de**	00000000**
00000790	prenume*********	000007cc	00000000	15	0	7	18118000**	00000000**	00000000**
000007cc	data-nasterii***	00000934	00000808	19	1	12	00000003**	00000000**	00000000**
00000808	an**************	00000844	0000076c	10	0	12	00088000**	00000000**	000000c8**
adresse	identificateur	frer-per	fils	typ	cmax	origine	tail ptpr	org adrdic	nbmoy
00000844	luna************	00000880	00000001	10	0	12	08048000**	00000000**	0000000b**
00000880	zi**************	000008bc	00000001	10	0	12	0c058000**	00000000**	0000001e**
000008bc	pers-locn*******	fffff834	00001ce4	7*	0	13	00000002**	00000026**	00001e88**

type	identificteur	cmax	tail
bloc	fichier	1	651006793
enti	persoana	36864	67
enti	judete	64	9
enti	localitati	16000	20
enti	grade-militare	128	8
enti	nationalitate-cetatenie	32	8
enti	functii	4096	19
enti	specialitati	4096	20
enti	copii	320000	74
enti	sanctiune	128	17
enti	sanctiuni	160000	3833
enti	profesii	128	9
enti	recompensa	128	18
enti	recompense	160000	69

```
%        DACTI TB:44
%        DACTI TB:64

%        ACTI TB:49

%        RUN FN:D

** socrate started **   v1.5    le: 17/11/87  a 18.05.10.64.
     d
     /*                                                      */
     /* descriere format date de intrare in sistem           */
     /* <<<     adaugare la structura               >>>      */
     /*                                                      */
     /*------------------------------------------------------*/
     /*   descrierea formatelor pentru incarcare-actualizare */
     /*        entitate : <persoana>                         */
     /*------------------------------------------------------*/
     formal pers
      debut
       ident             mot 3
       matricol          mot (13)
        /*---------------------------------------------------*/
        /*   redefinire componente logice                    */
        /*---------------------------------------------------*/
          sex            mot        (4  1)
          an-nastere     dilate     (5  2)
          luna-nastere   dilate     (7  2)
          zi-nastere     dilate     (9  2)
       nume              mot        (16  )
       prenume           mot         16
       /* < loc nastere >                                    */
         judet           dilate     ( 2 )
         localit         dilate       6
         cetatenie       mot          1
```

```
      nationalitate   mot        1
      stare-civila    mot        1
    /*  <apartenenta politica>                      */
      tip-apart       mot        4
      an-apart        dilate     4
      ll-apart        dilate     2
      zz-apart        dilate     2
    /* <livret-militar>                             */
      seria-livret    mot        3
      nr-livret       dilate     (   6    )
   /* <<<<<<  format p02 redefinire  p01            */
    /* situatia militara                            */
      specialit       dilate     (17  3)
      comisariat      dilate     (20 2 )
      grad            dilate     ( 22 2)
      obligatii       mot         (24  1)
      grupa-sanguina  mot        (25 4 )
    /* <buletin de identitate>                      */
      seria-bi        mot        29 2
      nr-bi           dilate     31 6
      an-emit-bi      dilate     37 4
      den1-circa      mot        (41 20)
      den2-circa      mot        (61 20)
  /*------------------------------------------------*/
  /* exercitiu: descrieti un format de intrare pentru */
  /* restul atributelor asociate entitati <persoana>  */
  /*------------------------------------------------*/
    fin
  fin
  ?
!typ !identificateur                  !cmax      !tail         !
!----!--------------------------------!----------!-------------!
!bloc!fichier                         ! 1        ! 651006793   !
!----!--------------------------------!----------!-------------!
!enti!persoana                        ! 36864    ! 67          !
!----!--------------------------------!----------!-------------!
!enti!judete                          ! 64       ! 9           !
!----!--------------------------------!----------!-------------!
!enti!localitati                      ! 16000    ! 20          !
!----!--------------------------------!----------!-------------!
!enti!grade-militare                  ! 128      ! 8           !
!----!--------------------------------!----------!-------------!
!enti!nationalitate-cetatenie         ! 32       ! 8           !
!----!--------------------------------!----------!-------------!
!enti!functii                         ! 4096     ! 19          !
!----!--------------------------------!----------!-------------!
!enti!specialitati                    ! 4096     ! 20          !
!----!--------------------------------!----------!-------------!
!enti!copii                           ! 320000   ! 74          !
!----!--------------------------------!----------!-------------!
!enti!sanctiune                       ! 128      ! 17          !
!----!--------------------------------!----------!-------------!
!enti!sanctiuni                       ! 160000   ! 3833        !
!----!--------------------------------!----------!-------------!
!enti!profesii                        ! 128      ! 9           !
!----!--------------------------------!----------!-------------!
!enti!recompensa                      ! 128      ! 18          !
!----!--------------------------------!----------!-------------!
!enti!recompense                      ! 160000   ! 69          !
!----!--------------------------------!----------!-------------!
!form!pers                            ! 1        ! 80          !
!----!--------------------------------!----------!-------------!
%       RUN FN:D
** socrate started **  v1.5   le: 17/11/87  a 18.05.18.30.
    d
    /* ------------------------------------------*/
    /*                                           */
```

```
/*            <<macnom>>                          */
/*               ''                               */
/* ----------        --------------------------- */
/*                                                */
/*    adaugare la structura formate               */
/*           <<nomenclatoare>>                    */
/* -------------------------------------------- */
formal              macheta
 - debut
  ident             mot 3
  filler            mot 77
  /*   <nationalitate-cetatenie>                  */
  cod-nac           mot (4  1)
  den1-nac          mot (5 25)
  /*      <grade-militare>                        */
  cod-grm           dilate (4  2)
  den1-grm          mot    (6 27)
  /*          <judete>                            */
  cod-jud           dilate (4  2)
  cifk              dilate (6  1)
  den1-jud          mot    (7 15)
  /*       <localitati>                           */
  ref1-jud          dilate (4  2)
  cod-loc           dilate (6  6)
   ref1-loc   dilate (12  6)
  den1-loc          mot    (18 30)
  den2-loc          mot    (48 10)
  /*          <profesii>                          */
  cod-pro           dilate (4  2)
  den1-pro          mot    (6 30)
  den2-pro          mot    (36 30)
  /*  <specialitati>                              */
  ref1-fun          dilate  (4  4)
  cod-spe           dilate  (8  4)
  den1-spe          mot        (12 30)
  den2-spe          mot        (42 30)
  /*          <sanctiuni>                         */
  cod-san           dilate  (4  3)
  den1-san          mot     (7  30)
  den2-san          mot     (37 30)
  /*          <functii                            */
  cod-fun           dilate  (4  4)
  den1-fun          mot     (8 30)
  den2-fun          mot     (38 20)
 fin
 /* sfirsit <bloc formal> <macheta>    */
fin
/* sfirsit <bloc> definire / adaugare <d> */
?

%        LOGOUT
0064  personal   an = 1010   ph = 0001   date = 17/11/87-321
h.deb = 18h 03m 57s   h.fin = 18h 05m 31s   time = 00003083   code = 000
lgp = 00110   mem = 00038   lo = 000960   in = 00442   out = 00000
         * APPROCHE FIN PROCEDURE CATALOGUEE
         EST                    FIN DE LA PROCEDURE CATALOGUEE
         EOJ
```

CATALOGAREA PROGRAMELOR PENTRU BAZA DE DATE

In exemplul nostru programele scrise in LMD au fost stocate sub forma unui fisier de comenzi , cu numele MACRO, in biblioteca PERSONAL. Se vor cataloga si programele generate automat care se regasesc in biblioteca sistem Z%PROC sub numele CAMIDENT.

```
.        * COMPILARE/CATALOGARE PROGRAME PENTRU BD
.        XPROC *,DVT:AD,VS:ADIAD1,VN:0,FN:'PERSONALSOU/MACRO',MOD
.        MOD 23,23
:defmac linast80 :exp
.        ENDMOD
.        XPROC PROCGEN,MOD                                          1
.        MOD &SUPB:DV:ADO&                                          2
.        ENDMOD                                                     3
.        FETCH LN:BIBIMTSO,GN:1,VN:1,FN:SOCBTCH,                  * 1
.             DV:ADO                                                2
.        ASSIGN S,DVT:AD,VS:ADIAD1                                  3
.        ASSIGN Q,DVT:AD,VS:ADIAD1                                  4
.        LABEL S,AM:OFL,FN:'BAZA DE BAZE'                           5
.        LABEL Q,AM:OFL,FN:'MANEVRA SGBD'                           6
.        * ASSIGN Z,DV:MT00                                         7
.        * LABEL Z,FN:'JURNAL'                                      8
.        ASSIGN T,DV:AD7                                            9
.        * APPROCHE FIN PROCEDURE CATALOGUEE
.        EST                    FIN DE LA PROCEDURE CATALOGUEE
.        OPTION CS'CF:50,DS:4096'                                   4
.        RUN TIME:999,NL:500000,AD:0,0,KEY:60                       5
socbtch  started
nb cadre fich: 50
%        SOC BN:BDDPERS,PN:AVRAM,AN:0001,PW:SOC                     6
%        RUN FN:M,LST                                               7
** socrate started **  v1.5    le: 17/11/87  a 18.06.23.02.
   :defpro pageskip :exp i ' ' :fdef?                               8
ok prog comp enregistre  pageskip
%        RUN FN:M,LST                                               9
** socrate started **  v1.5    le: 17/11/87  a 18.06.28.34.
   :supexp linast120 ?                                             10
err. macro 301 nl 1  .. linast120
erreur en mode macro
%        RUN FN:M,LST                                              11

** socrate started **  v1.5    le: 17/11/87  a 18.06.28.94.
   :defmac linast120 :exp                                          12
       i ( 1) '********************************'                   13
       i (+0) '********************************'                   14
       i (+0) '********************************'                   15
       i (+0) '********************************'                   16
         ecrire                                                    17
   :fdef?                                                          18
ok  macro  enregistree : linast120
%        RUN FN:M,LST                                              19
** socrate started **  v1.5    le: 17/11/87  a 18.06.29.90.
   :supexp linast80                                                20
   ?                                                               21
err. macro 301 nl 1  .. linast80
erreur en mode macro
%        RUN FN:M,LST                                              22
** socrate started **  v1.5    le: 17/11/87  a 18.06.30.84.
   :defmac linast80 :exp
   i (1) '********************************'                        24
     i(+0) '********************************'                      25
     i(+0) '*********************'                                 26
         ecrire                                                    27
   :fdef?                                                          28
```

```
ok  macro  enregistree : linast80
%       RUN FN:M,LST
** socrate started **  v1.5    le: 17/11/87  a 18.06.31.92.
 1    :supexp capnom          ?
err. macro 301 nl 1  .. capnom
erreur en mode macro
%       RUN FN:M,LST
** socrate started **  v1.5    le: 17/11/87  a 18.06.32.88.
 1    :defmac
 2     capnom
 3    :exp
 4    i ' ' m y2 = 0
 5     si y1 > 0
 6      alors
 7       i (71) 'pagina:' i (80 -4) y1 ecrire
 8       m y2 = y2 + 1
 9     fin
10     linast80
11     i (1) '*  c o d *' i (30) 'd e n u m i r e ' i(80) '*'
12     ecrire
13     linast80
14    m y2 = y2 + 4
15    :fdef?
ok  macro  enregistree : capnom
%       RUN FN:M,LST
** socrate started **  v1.5    le: 17/11/87  a 18.06.36.12.
 1    :supexp  lisnom         ?
err. macro 301 nl 1  .. lisnom
erreur en mode macro
%       RUN FN:M,LST
** socrate started **  v1.5    le: 17/11/87  a 18.06.37.24.
 1    :defmac
 2     lisnom : denumit : cu : rinduri pe pagina
 3     d x1
 4    :exp
 5    faire
 6    d x1
 7     exec pageskip
 8      i ' ' i (15) 'nomenclatorul de' i (+1) ':2:' ecrire
 9     capnom
10     m y2 = y2 + 2
11     m y3 = :3: m y3 = y3 - 1
12     pour tout :1: x1
13      si y2 >= y3
14       alors
15        m y1 = y1 + 1
16        exec pageskip
17        capnom
18      fin
19      i (1) '*'
20      i (9 -7) cod
21      i (10) '*'
22      i (12) den1
23      i (+0) den2
24      i (80) '*'
25       ecrire
26      linast80
27      m y2 = y2 + 2
28      d x1
29    fin
30     fin
31    :fdef?
ok  macro  enregistree : lisnom
%       RUN FN:M,LST
** socrate started **  v1.5    le: 18/11/87  a 18.06.41.94.
 1    :supexp cartela?
```

```
%       RUN FN:M,LST
** socrate started **  v1.5    le: 17/11/87  a 18.06.42.64.
 1   :defmac cartela :exp macheta :fdef?
ok  macro  enregistree : cartela
%       RUN FN:M,LST
** socrate started **  v1.5    le: 17/11/87  a 18.06.43.56.
 1   :supexp act-p01 ?
err. macro 301 nl 1  .. act-p01
erreur en mode macro
%       RUN FN:M,LST
** socrate started **  v1.5    le: 17/11/87  a 18.06.44.18.
 1   :defpro  act-p01
 2   :contxt  d x2 = un persoana /* x2 = parametru de apel       */
 3   :exp  m x4 = u
 4     m x1 = formal pers dans buf 1
 5     pour x2  /* citatiile din<stinga> in context<persoana>    */
 6     m z1 = u m z1 = nume de x1
 7      si existe z1 /* modif,conditionala datorata dublului     */
 8        alors      /* rol al machetei<<creere-actualizare>>    */
 9         m nume = u              /* daca..nume de x1..         */
10         m nume = nume de x1    /* este spatiu                */
11        fin                     /* z1 ramine la               */
12       m z1 = u m z1 = prenume de x1     /*  valoarea u       */
13       si existe z1             /* ===========================*/
14         alors
15          m prenume = u
16          m prenume = prenume de x1
17       fin
18       si pas an de data-nasterii
19         alors m y1 = u
20        m y1 = an-nastere de x1
21          m y1 = y1 + 1900
22          m an de data-nasterii = y1
23          m luna de data-nasterii = luna-nastere de x1
24          m zi de data-nasterii = zi-nastere de x1
25       fin
26       /* incarcare refere <<pers-locn>> */
27       m y1 = u m y1 = localit de x1
28       m x4 = u m x4 = un localitati avec cod = y1;
29       si existe x4
30         alors
31          m pers-locn de data-nasterii = u /*evit.incoerenta*/
32          m pers-locn de data-nasterii = x4
33       fin
34       m z1 = u m z1 = cetatenie de x1
35       si existe z1
36         alors
37          m cetatenie = u
38          m cetatenie = cetatenie de x1
39       fin
40       m z1 = u m z1 = nationalitate de x1
41       si existe z1
42         alors
43          m nationalitate = u
44          m nationalitate = nationalitate de x1
45       fin
46       m z1 = u m z1 = stare-civila de x1
47       si existe z1
48         alors
49 .        m stare-civila de x2 = u /* contextul<x2> implicit*/
50          m stare-civila = stare-civila de x1
51       fin
52       m z1 = u m z1 = tip-apart de x1
53       si existe z1
54         alors
55          m apartenenta = u
```

```
56        m apartenenta = tip-apart de x1                                    140
57        m an   de data-pcr = an-apart de x1                                141
58    m zi de data-pcr = zz-apart de x1
59        m luna de data-pcr = ll-apart de x1                                143
60      fin                                                                  144
61      m z1 = u m z1 = seria-livret de x1                                   145
62      si existe z1                                                         146
63        alors                                                              147
64        m seria de livret de situatia-militara =                           148
65                          seria-livret de x1                               149
66        m nr   de livret de situatia-militara =                            150
67                          nr-livret de x1                                  151
68      fin                                                                  152
69     fin /* pentru <pour x2> */                                            153
70   :fdef                                                                   154
71        ?                                                                  155
ok prog comp enregistre  act-p01
%       RUN FN:M,LST                                                         156
** socrate started **  v1.5    le: 17/11/87  a 18.07.07.42.
 1   :supexp act-p02 ?                                                       157
err. macro 301 nl 1  .. act-p02
erreur en mode macro
%       RUN FN:M,LST                                                         158
** socrate started **  v1.5    le: 17/11/87  a 18.07.08.18.
 1   :defpro act-p02                                                         159
 2   :contxt d x2 = un persoana                                              160
 3   :exp m x1 = formal pers dans buf 1                                      161
 4     pour x2                                                               162
 5      i (1  )          ' scrieti programul pentru'                         163
 6      i (+1 )          !tratarea machetei<p02>;'                           164
 7      i (+5 )          !eventual modificati programul'                     165
 8      i (+0 )          'principal pentru a trata'                          166
 9        ecrire                                                             167
10      i (2  30)        'si macheta<p03>;scrieti secventa'                  168
11      i (+0 )          'de program corespondenta.'                         169
12          ecrire    i '?(mult succes)?'                                    170
13      i (1    ) '<marca>;'  m z2 = matricol de x1 /* inutil */             171
14      i (+15 -13) z2  i (+1 ) '<nume>;' i (+0 16) nume ecrire              172
15     fin                                                                   173
16   :fdef ?                                                                 174
warning er*quot*nl 10  si macheta<p03>;scrieti secven
ok prog comp enregistre  act-p02
%       RUN FN:M,LST                                                         175
** socrate started **  v1.5    le: 17/11/87  a 18.07.15.00.
 1   :lisall ?                                                               176
c pageskip                          1
m linast120                         1
m linast80                          1
m capnom                            2
        m linast80
m lisnom                            4
        c pageskip
        m capnom
        m linast80
m cartela                           1
c act-p01                           4
c act-p02                           2
** fin liste **
%       LOGOUT                                                               177
0065  personal   an = 1010   ph = 0001  .date = 17/11/87-321
h.deb = 18h 05m 34s   h.fin = 18h 07m 23s   time = 00002322   code = 000
lgp = 00110   mem = 00038   lo = 000280   in = 00231   out = 00000
        * APPROCHE FIN PROCEDURE CATALOGUEE
        EST                              FIN DE LA PROCEDURE CATALOGUEE
        EOJ
```

```
.        JOB PERSONAL,AN:1010,PN:SOC
.        * CATALOGARE MACROINSTRUCTIUNI
.        XPROC *,DV:ADO,GN:1,VN:0,FN:'Z%PROC  SOU/CAMIDENT',MOD
.        MOD 1,1,CN:7
.        XPROC PROCGEN,MOD
.        MOD &SUPB:DV:ADO&
.        ENDMOD
.        OPTION CS'CF:10,DS:6096'
.        RUN TIME:999,NL:500000,AD:0,0,KEY:60
%        SOC BN:BDDPERS,PN:AVRAM,PW:SOC,AN:0001
%        EVAL
&%&        run fn:m,&lst:lst&
.        ENDMOD
.        XPROC PROCGEN,MOD
.        MOD &SUPB:DV:ADO&
.        ENDMOD
.        FETCH LN:BIBIMTSO,GN:1,VN:1,FN:SOCBTCH,                    1
.                  DV:ADO                                           2
.        ASSIGN S,DVT:AD,VS:AD1AD1                                  3
.        ASSIGN Q,DVT:AD,VS:AD1AD1                                  4
.        LABEL S,AM:OFL,FN:'BAZA DE BAZE'                           5
.        LABEL Q,AM:OFL,FN:'MANEVRA SGBD'                           6
.        * ASSIGN Z,DV:MT00                                         7
.        * LABEL Z,FN:'JURNAL'                                      8
.        ASSIGN T,DV:AD7                                            9
.        * APPROCHE FIN PROCEDURE CATALOGUEE
.        EST                          FIN DE LA PROCEDURE CATALOGUEE
.        OPTION CS'CF:10,DS:6096'
.        RUN TIME:999,NL:500000,AD:0,0,KEY:60
socbtch  started
nb cadre fich: 10
%        SOC BN:BDDPERS,PN:AVRAM,PW:SOC,AN:0001
%        EVAL
%        RUN FN:M,LST
** socrate started **  v1.5  le: 17/11/87  a 18.07.52.30.
 1    :supexp cam-jud  ?                                            2
err. macro 301 nl 1  .. cam-jud
erreur en mode macro
%        RUN FN:M                                                   3
** socrate started **  v1.5  le: 17/11/87  a 18.07.52.86.
 1     :defmac                                                      4
 2      cam-jud                                                     5
 3     :exp                                                         6
 4      d  x1  d  x2  d  x3  d x4   m y6 = 0   m y7 = 0             7
 5      m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0           8
 6      m x1 = formal cartela  dans  buf  0                         9
 7      faire                                                      10
 8       lire dans buf 0                                           11
 9       m z1 = ident      de        x1                            12
10       si z1 = '¤¤¤¤'          alors  sortie  fin                13
11       m  y4 = y4 + 1                                            14
12       si z1 ^=  'jud'                                           15
13         alors                                                   16
14          m y7 = y7 + 1                                          17
15          i '?? ident ?? *eronat* '                              18
16          refaire                                                19
17      fin                                                        20
18      m z5 = u   m y1 = u                                        21
19      m y1 = cod-jud   de   x1                                   22
20      m x2 = un  judete  avec cod = y1 ;                         23
21     si existe x2                                                24
22       alors                                                     25
23        m z2 = denl-jud                   de x1                  26
24        si z2 = '  '                                             27
25          alors                                                  28
26           m denl      de  x2    = u                             29
27           m cod       de  x2    = u                             30
```

```
28              s x2
29              m y2 = y2 + 1    /* stergeri  */
30            sinon
31              m den1      de  x2    = den1-jud     de  x1
32              m y6 = y6 + 1    /* modificari(actualizari) */
33          fin
34        sinon
35         g  un  judete         x3
36         m  cod           de  x3    = cod-jud      de  x1
37         m den1           de  x3    = den1-jud    de x1
38         m  y3 = y3 + 1   /*  adaugari(creeri)   */
39        fin
40      refaire
41      fin
42      i (  1  ) ' ** au fost citite : '
43      i (+10 -10)  y4
44      i (+1     )       ' inregistrari < jud      > '
45      i (+0     )       ' din care: '
46      ecrire
47      i (+20 -10)  y3
48      i (+1     )       ' adaugari(creeri); '
49      ecrire
50      i (+20 -10)  y2
51      i (+1     )       ' stergeri(suprimari); '
52      ecrire
53      i (+20 -10) y6
54      i (+1     )       ' modificari(actualizari); '
55      ecrire
56      i (+20 -10)  y7
57      i (+1     )       ' eronate  '
58      ecrire
59    :fdef  ?
ok  macro  enregistree : cam-jud
%       RUN FN:M
** socrate started **   v1.5   le: 17/11/87   a 18.07.59.04.
 1    :supexp  cam-loc   ?
err. macro 301 nl 1  .. cam-loc
erreur en mode macro
%       RUN FN:M
** socrate started **   v1.5  le: 17/11/87   a 18.08.00.56.
 1    :defmac
 2      cam-loc
 3    :exp
 4     d  x1  d   x2   d  x3  d x4    m y6 = 0    m y7 = 0
 5     m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
 6     m x1 = formal cartela  dans  buf  0
 7     faire
 8      lire dans buf 0
 9      m z1 = ident       de        x1
10      si z1 = '¤¤¤'            alors  sortie  fin
11      m  y4 = y4 + 1
12      si z1 ^=  'loc'
13        alors
14          m y7 = y7 + 1
15          i '?? ident ?? *eronat* '
16          refaire
17      fin
18      m z5 = u    m y1 = u
19      m y1 = cod-loc   de    x1
20      m x2 = un  localitati  avec cod = y1 ;
21    si existe x2
22      alors
23      m z2 = den1-loc                 de  x1
24      si z2 = '  '
25        alors
26          m den1      de  x2    = u
```

```
         m cod         de  x2   = u
         m den2        de  x2   = u
         m y2 = y2 + 1   /* stergeri */
        sinon
         m den1        de  x2   = den1-loc       de  x1
         m den2        de  x2   = den2-loc       de  x1
         m y6 = y6 + 1   /* modificari(actualizari) */
        fin
      sinon
       g  un  localitati        x3
       m  cod          de  x3   = cod-loc        de  x1
       m  den1         de  x3   = den1-loc       de  x1
       m  den2         de  x3   = den2-loc       de  x1
       m y3 = y3 + 1          /* adaugari(creeri)  */
      fin
      m x2 = un localitati          avec  cod = y1 ;
     m y5 = ref1-loc  de x1
      m x4 = un  localitati      avec  cod  = y5 ;
      m loc-loc  de  x2  =  x4
     refaire
     fin
     i (  1 ) ' ** au fost citite : '
     i (+10 -10) y4
     i (+1      )      ' inregistrari < loc > '
     i (+0      )      ' din care: '
     ecrire
     i (+20 -10)  y3
     i (+1      )     ' adaugari(creeri); '
     ecrire
     i (+20 -10)  y2
     i (+1      )     ' stergeri(suprimari); '
     ecrire
     i (+20 -10) y6
     i (+1      )     ' modificari(actualizari); '
     ecrire
     i (+20 -10) y7
     i (+1      )     ' eronate  '
     ecrire
    :fdef ?
ok  macro  enregistree : cam-loc
%       RUN FN:M
** socrate started **   v1.5    le: 17/11/87   a 18.08.13.70.
    :supexp  cam-spe   ?
err. macro 301 nl 1  .. cam-spe
erreur en mode macro
%       RUN FN:M
** socrate started **   v1.5    le: 17/11/87   a 18.08.14.46.
     :defmac
     cam-spe
    :exp
     d  x1  d  x2  d  x3  d x4    m y6 = 0   m y7 = 0
    m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
    m x1 = formal cartela  dans  buf 0
    faire
     lire dans buf 0
     m z1 = ident     de      x1
     si z1 = '¤¤¤¤'          alors  sortie  fin
     m  y4 = y4 + 1
     si z1 ^=  'spe'
      alors
       m y7 = y7 + 1
       i '?? ident ?? *eronat* '
       refaire
     fin
     m z5 = u   m y1 = u
     m y1 = cod-spe   de    x1
     m x2 = un  specialitati  avec cod = y1 ;
```

```
      si existe x2
        alors
        m z2 = denl-spe              de  x1
        si z2 = ' '
          alors
           m denl        de x2   = u
           m cod .       de x2   = u
           m den2        de  x2   = u
           m y2 = y2 + 1    /* stergeri */
          sinon
           m denl        de x2    = denl-spe       de  x1
           m den2        de x2    = den2-spe       de  x1
           m y6 = y6 + 1 /* modificari(actualizari) */
         fin
       sinon
        g  un  specialitati          x3
        m  cod           de  x3    = cod-spe        de  x1
        m  denl          de  x3    = denl-spe       de  x1
        m  den2          de  x3    = den2-spe       de  x1
        m y3 = y3 + 1             /*  adaugari(creeri)  */
       fin
       m x2 = un specialitati           avec  cod = y1 ;
      m y5 = refl-spe  de x1
       m x4 = un  functii        avec  cod  = y5 ;
       m spe-fun  de  x2   =   x4
      refaire
      fin
      i  (  1  ) ' ** au fost citite : '
      i  (+10 -10)  y4
      i  (+1     )      ' inregistrari < spe      > '
      i  (+0     )      ' din care: '
       ecrire
       i  (+20 -10)  y3
       i  (+1     )     ' adaugari(creeri); '
       ecrire
       i  (+20 -10)  y2
       i  (+1     )     ' stergeri(suprimari); '
       ecrire
       i  (+20 -10)  y6
       i  (+1     )     ' modificari(actualizari); '
       ecrire
       i  (+20 -10)  y7
       i  (+1     )     ' eronate  '
       ecrire
     :fdef  ?
ok  macro  enregistree : cam-spe
%       RUN FN:M
** socrate started **  v1.5    le: 17/11/87  a 18.08.23.88.
    :supexp  cam-pro  ?
err. macro 301 nl 1  .. cam-pro
erreur en mode macro
%       RUN FN:M
** socrate started **   v1.5    le: 17/11/87  a 18.08.24.62.
      :defmac
       cam-pro
      :exp
       d  x1  d   x2  d   x3  d x4     m y6 = 0    m y7 = 0
       m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
       m x1 = formal cartela  dans  buf  0
       faire
        lire dans buf 0
        m z1 = ident         de          x1
        si z1 = 'xxxx'           alors  sortie  fin
        m   y4 = y4 + 1
        si z1 ^= 'pro'
          alors
           m y7 = y7 + 1
```

```
        i '?? ident ?? *eronat* '
        refaire
      fin
      m z5 = u   m yl = u
      m yl = cod-pro   de    x1
      m x2 = un  profesii  avec cod = yl ;
    si existe x2
      alors
        m z2 = denl-pro                    de x1
        si z2 = '  '
          alors
            m denl      de  x2   = u
            m cod       de  x2   = u
            s x2
            m y2 = y2 + 1   /* stergeri  */
          sinon
            m denl      de  x2   = denl-pro     de  x1
            m y6 = y6 + 1   /* modificari(actualizari) */
        fin
      sinon
       g  un  profesii          x3
       m  cod              de  x3   = cod-pro      de  x1
       m denl              de  x3   = denl-pro   de x1
       m   y3 = y3 + 1  /*  adaugari(creeri)  */
      fin
    refaire
    fin
    i (  1  ) ' ** au fost citite : '
    i (+10 -10)  y4
    i (+1      )       ' inregistrari < pro     > '
    i (+0      )       ' din care: '
     ecrire
    i (+20 -10)  y3
    i (+1      )       ' adaugari(creeri); '
     ecrire
    i (+20 -10)  y2
    i (+1      )       ' stergeri(suprimari); '
     ecrire
    i (+20 -10) y6
    i (+1      )       ' modificari(actualizari); '
     ecrire
    i (+20 -10) y7
    i (+1      )       ' eronate  '
     ecrire
    :fdef  ?
ok  macro  enregistree : cam-pro
%       RUN FN:M
** socrate started **  v1.5     le: 17/11/87   a 18.08.30.72.
     :supexp  cam-fun  ?
err. macro 301 nl 1  .. cam-fun
erreur en mode macro
%       RUN FN:M
** socrate started **  v1.5     le: 17/11/87   a 18.08.31.82.
     :defmac
      . cam-fun
     :exp
      d  x1  d  x2  d  x3  d x4    m y6 = 0   m y7 = 0
      m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
      m x1 = formal cartela  dans  buf  0
      faire
       lire dans buf 0
       m z1 = ident        de          x1
       si z1 = '¤¤¤¤'          alors  sortie  fin
       m  y4 = y4 + 1
       si z1 ^= 'fun'
         alors
           m y7 = y7 + 1
```

```
         i '?? ident ?? *eronat* '
         refaire
      fin
      m z5 = u   m y1 = u
      m y1 = cod-fun   de   x1
      m x2 = un  functii  avec cod = y1 ;
    si exista x2
       alors
       m z2 = den1-fun                de  x1
       si z2 = '  '
         alors
          m den1      de  x2    = u
          m cod       de  x2    = u
          m den2      de  x2    = u
          m y2 = y2 + 1   /* stergeri */
         sinon
          m den1      de  x2    = den1-fun     de  x1
          m den2      de  x2    = den2-fun     de  x1
          m y6 = y6 + 1   /* modificari(actualizari) */
       fin
     sinon
      g  un  functii         x3
      m  cod          de  x3    = cod-fun      de  x1
      m  den1         de  x3    = den1-fun     de  x1
      m  den2         de  x3    = den2-fun     de  x1
      m y3 = y3 + 1          /*  adaugari(creeri)  */
     fin
    refaire
    fin
    i (  1  ) ' ** au fost citite : '
    i (+10 -10)  y4
    i (+1     )       ' inregistrari < fun     > '
    i (+0     )       ' din care: '
     ecrire
    i (+20 -10)  y3
    i (+1     )       ' adaugari(creeri); '
     ecrire
    i (+20 -10)  y2
    i (+1     )       ' stergeri(suprimari); '
     ecrire
    i (+20 -10) y6
    i (+1     )       ' modificari(actualizari); '
     ecrire
    i (+20 -10) y7
    i (+1     )       ' eronate  '
     ecrire
    :fdef  ?
ok  macro  enregistree : cam-fun
%         RUN FN:M
** socrate started **  v1.5    le: 17/11/87  a 18.08.40.20.
   :supexp  cam-san  ?
err. macro 301 nl 1  .. cam-san
erreur en mode macro
%         RUN FN:M
** socrate started **   v1.5    le: 17/11/87  a 18.08.41.60.
    :defmac
     cam-san
    :exp
     d  x1  d  x2  d  x3  d x4     m y6 = 0   m y7 = 0
     m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0   m y5 = 0
     m x1 = formal cartela  dans  buf  0
     faire
      lire dans buf 0
      m z1 = ident       de          x1
      si z1 = '¤¤¤¤'          alors  sortie  fin
      m  y4 = y4 + 1
      si z1 ^=  'san'
```

```
13        alors
14         m y7 = y7 + 1
15         i '?? ident ?? *eronat* '
16         refaire
17       fin
18       m z5 = u   m y1 = u
19       m y1 = cod-san   de   x1
20       m x2 = un  sanctiune  avec cod = y1 ;
21      si existe x2
22        alors
23        m z2 = den1-san                    de  x1
24        si z2 = '  '
25          alors
26           m den1        de  x2   = u
27           m cod         de  x2   = u
28           m den2        de  x2   = u
29           m y2 = y2 + 1    /* stergeri */
30          sinon
31           m den1        de  x2    = den1-san         de  x1
32           m den2        de  x2    = den2-san         de  x1
34        fin
35       sinon
36        g  un  sanctiune         x3
37        m  cod           de  x3    = cod-san          de  x1
38        m  den1          de  x3    = den1-san         de  x1
39        m  den2          de  x3    = den2-san         de  x1
40        m y3 = y3 + 1     /*  adaugari(creeri)   */
41       fin
42      refaire
43      fin
44      i (  1  ) ' ** au fost citite : '
45      i (+10 -10)  y4
46      i (+1      )        ' inregistrari < san     > '
47      i (+0      )        ' din care: '
48      ecrire
49      i (+20 -10)  y3
50      i (+1      )        ' adaugari(creeri); '
51      ecrire
52      i (+20 -10)  y2
53      i (+1      )        ' stergeri(suprimari); '
54      ecrire
55      i (+20 -10)  y6
56      i (+1      )        ' modificari(actualizari); '
57      ecrire
58      i (+20 -10)  y7
59      i (+1      )        ' eronate '
60      ecrire
61     :fdef  ?
ok  macro  enregistree : cam-san
%       RUN FN:M
** socrate started **  v1.5    le: 17/11/87  a 18.08.49.94.
 1   :supexp  cam-rec  ?
err. macro 301 nl 1  .. cam-rec
erreur en mode macro
%       RUN FN:M
** socrate started **  v1.5    le: 17/11/87  a 18.08.51.46.
 1     :defmac
 2      cam-rec
 3     :exp
 4      d  x1  d  x2  d  x3  d x4    m y6 = 0   m y7 = 0
 5      m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
 6      m x1 = formal cartela  dans  buf 0
 7      faire
 8       lire dans buf 0
 9       m z1 = ident       de          x1
10       si z1 = 'xxxx'          alors  sortie  fin
11       m  y4 = y4 + 1
```

```
12     si z1 ^= 'rec'
13       alors
14       m y7 = y7 + 1
15       i '?? ident ?? *eronat* '
16       refaire
17     fin
18     m z5 = u   m y1 = u
19     m y1 = cod-rec    de    x1
20     m x2 = un  recompensa  avec cod = y1 ;
21    si existe x2
22       alors
23       m z2 = den1-rec          de  x1
24       si z2 = ' '
25         alors
26          m den1       de  x2   = u
27          m cod        de  x2   = u
28          m den2       de  x2   = u
29          m y2 = y2 + 1   /* stergeri */
30         sinon
31          m den1       de  x2    = den1-rec      de  x1
32          m den2       de  x2    = den2-rec      de  x1
33          m y6 = y6 + 1   /* modificari(actualizari) */
34       fin
35     sinon
36      g  un  recompensa          x3
37      m  cod           de  x3    = cod-rec       de  x1
38      m  den1          de  x3    = den1-rec      de  x1
39      m  den2          de  x3    = den2-rec      de  x1
40      m y3 = y3 + 1           /*  adaugari(creeri)  */
41     fin
42    refaire
43    fin
44     i ( 1  ) ' ** au fost citite : '
45     i (+10 -10) y4
46     i (+1     )      ' inregistrari < rec      > '
47     i (+0     )      ' din care: '
48     ecrire
49     i (+20 -10) y3
50     i (+1     )      ' adaugari(creeri); '
51     ecrire
52     i (+20 -10) y2
53     i (+1     )      ' stergeri(suprimari); '
54     ecrire
55     i (+20 -10) y6
56     i (+1     )      ' modificari(actualizari); '
57     ecrire
58     i (+20 -10) y7
59     i (+1     )      ' eronate  '
60     ecrire
61  :fdef  ?
ok  macro, enregistree : cam-rec
%        RUN FN:M
** socrate started **  v1.5    le: 17/11/87   a 18.09.01.00.
 1   :supexp  cam-grm  ?
err. macro 301 nl 1  .. cam-grm
erreur en mode macro
%        RUN FN:M
** socrate started **  v1.5    le: 17/11/87   a 18.09.03.76.
 1     :defmac
 2      cam-grm
 3     :exp
 4      d  x1   d  x2  d  x3  d x4    m y6 = 0   m y7 = 0
 5      m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
 6      m x1 = formal cartela  dans  buf  0
 7      faire
 8       lire dans buf 0
 9       m z1 = ident      de        x1
```

```
    si z1 = 'xxxx'          alors  sortie  fin
    m  y4 = y4 + 1
    si z1 ^=  'grm'
      alors
        m y7 = y7 + 1
        i '?? ident ?? *eronat* '
      *refaire
    fin
    m z5 = u   m y1 = u
    m y1 = cod-grm    de    x1
    m x2 = un grade-militare  avec cod = y1 ;
  si existe x2
      alors
      m z2 = den1-grm                   de x1
        si z2 = ' '
        alors
          m den1     de  x2    = u
          m cod      de  x2    = u
          s x2
          m y2 = y2 + 1    /* stergeri  */
        sinon
          m den1     de  x2    = den1-grm    de  x1
          m y6 = y6 + 1    /* modificari(actualizari) */
      fin
    sinon
      g  un  grade-militare          x3
      m  cod          de  x3   = cod-grm      de  x1
      m den1          de  x3   = den1-grm   de x1
      m  y3 = y3 + 1   /* adaugari(creeri)   */
    fin
   refaire
   fin
   i (  1  ) ' ** au fost citite : '
   i (+10 -10)  y4
   i (+1     )          ' inregistrari < grm    > '
   i (+0     )    ' din care: '
    ecrire
   i (+20 -10)  y3
   i (+1     )     ' adaugari(creeri); '
    ecrire
   i (+20 -10) y2
   i (+1     )      ' stergeri(suprimari); '
    ecrire
   i (+20 -10) y6
   i (+1       )     ' modificari(actualizari); '
    ecrire
   i (+20 -10) y7
   i (+1       )     ' eronate '
    ecrire
   :fdef  ?
ok  macro  enregistree : cam-grm
%       LOGOUT
0066  personal   an = 1010   ph = 0001   date = 17/11/87-321
h.deb = 18h 07m 25s   h.fin = 18h 09m 33s   time = 00002651   code = 000
lgp = 00110   mem = 00038   lo = 000608   in = 00561   out = 00000
.       ** O!K!<MACRO>
.       * APPROCHE FIN PROCEDURE CATALOGUEE
.       EST                                FIN DE LA PROCEDURE CATALOGUEE
.       EOJ PERSONAL,AN:1010,PN:,SOC
```

EXPLOATAREA SI INTRETINERA BAZEI DE DATE

Se dau exemple de apel al prgramelor catalogate in spatiul bazei de date si exemple de programe de exploatare necatalogate.

```
.       JOB PERSONAL,AN:1010,PN:SOC
.       XPROC PROCGEN,MOD
.       MOD &SUPB:DV:ADO&
.       ENDMOD
.       FETCH LN:BIBIMTSO,GN:1,VN:1,FN:SOCBTCH,                    *     1
.              DV:ADO                                                    2
.       ASSIGN S,DVT:AD,VS:AD1AD1                                        3
.       ASSIGN Q,DVT:AD,VS:AD1AD1                                        4
.       LABEL S,AM:OFL,FN:'BAZA DE BAZE'                                 5
.       LABEL Q,AM:OFL,FN:'MANEVRA SGBD'                                 6
.       * ASSIGN Z,DV:MT00                                               7
.       * LABEL Z,FN:'JURNAL'                                            8
.       ASSIGN T,DV:AD7                                                  9
.       * APPROCHE FIN PROCEDURE CATALOGUEE
.       EST                          FIN DE LA PROCEDURE CATALOGUEE)
.       OPTION CS'CF:50,DS:4096'
.       RUN TIME:999,NL:500000,AD:0,0,KEY:60
socbtch  started
nb cadre fich: 50
%       SOC BN:BDDPERS,PN:AVRAM,PW:SOC,AN:0001
%       EVAL
%       RUN FN:M
** socrate started **  v1.5   le: 17/11/87  a 18.17.14.46.
              1    :supexp  lisnom          ?
ok exp macro supprimee : lisnom
%       RUN FN:M
** socrate started **  v1.5   le: 17/11/87  a 18.17.29.20.
              1    :defmac
              2      lisnom : denumit : cu : rinduri pe pagina
              3    :exp
              4    faire
              5    d x1
              6     exec pageskip
              7      i ' ' i (15) 'nomenclatorul de' i (+1) ':2:' ecrire
              8      capnom
              9      m y2 = y2 + 2
             10      m y3 = :3: m y3 = y3 - 1
             11      pour tout :1: x1
             12       si y2 >= y3
             13        alors
             14          m y1 = y1 + 1
             15          exec pageskip
             16          capnom
             17       fin
             18       i (1) '*'
             19       i (9 -7) cod
             20       i (10) '*'
             21       i (12) den1
             22       i (+0) den2
             23       i (80) '*'
             24        ecrire
             25      linast80
             26      m y2 = y2 + 2
             27     fin
             28    :fdef?
ok  macro  enregistree : lisnom
%       RUN FN:R
** socrate started **  v1.5   le: 17/11/87  a 18.17.44.20.
              1         cam-jud ?
1: 420/ 918/ 498
```

```
d: 23/ 646
                2    jud010alba
                3    jud021arad
                4    jud031arges
                5    jud403municipiul bucuresti
                6    ¤¤¤¤
** au fost citite :          4  inregistrari < jud    >  din care:
                4  adaugari(creeri);
                0  stergeri(suprimari);
                0  modificari(actualizari);
                0  eronate
t: 21/ 190/ 169
d: 2/ 41
%       RUN FN:M
** socrate started **  v1.5   le: 17/11/87  a 18.17.57.54.
                1    :supexp cam-loc?
ok exp macro supprimee : cam-loc
%       RUN FN:M
** socrate started **  v1.5   le: 17/11/87  a 18.17.59.66.
                1    :edimac cam-loc?
                2    45 sa /loc-loc/sup-loc/
 45
   m sup-loc  de  x2 =  x4
                3    *
ok  macro  enregistree : cam-loc
%       RUN FN:R
** socrate started **  v1.5    le: 17/11/87  a 18.18.10.58.
                1       cam-loc ?
t: 562/ 1296/ 734
d: 28/ 1084
                2    loc03000465000000domnesti
                3    loc03000466000465stanesti
                4    loc03000467000465pitrosani
                5    loc03000468000467varzaroaia
                6    loc40007000000000bucuresti
                7    ¤¤¤¤
 ** au fost citite :          5  inregistrari < loc    >  din care:
                5  adaugari(creeri);
                0  stergeri(suprimari);
                0  modificari(actualizari);
                0  eronate
t: 41/ 194/ 153
d: 11/ 67
%       RUN FN:R
** socrate started **  v1.5   le: 17/11/87  a 18.18.28.36.
                1     i
                2    pour tout localitati x1
                3          ayant pas sup-loc;
                4      i (5 -5) cod       i (+1) den1 i (+0) den2 ecrire
                5       pour tout loc-sup x1
                6       i (10 -5) cod
                7                         i (+1) den1 i (+0) den2 ecrire
                8       pour tout loc-sup x1
                9       i (15 -5) cod
                10                        i (+1) den1 i (+0) den2 ecrire
                11      pour tout loc-sup x1
                12      i (20 -5) cod
                13                        i (+1) den1 i (+0) den2 ecrire
                14      pour tout loc-sup x1
                15      i (25 -5) cod
                16                        i (+1) den1 i (+0) den2 ecrire
                17          si existe sup-loc
                18            alors
                19          i 'err.ff. eroare fatala de date?'
                20            sortie
                21        fin
                22   fin fin fin fin fin ?
```

```
t: 327/ 688/ 361
d: 0/ 589
 465 domnesti
 466 stanesti
 467 pitrosani
 468 varzaroaia
7000 bucuresti
t: 14/ 36/ 22
d: 0/ 35
%        RUN FN:0
** socrate started **  v1.5    le: 17/11/87  a 18.18.38.86.
             1    cam-san  ?
t: 464/ 1208/ 744
d: 27 922
             2    san014excludere
             3    san013vot de blam cu avertisment
             4    san012vot de blam
             5    san011mustrare
             6    san008inloc din func-trece alta munc
             7    san007retragere 1-6luni din functie
             8    san006desf disciplin contract munca
             9    san005retrograd 1clasa/categ 1-3luni
            10    san004dimin retrib-5-10% 1-3luni
            11    san003retragerea 1 gradatii/trepte
            12    san002avertisment
            13    san001mustrare
            14    ¤¤¤¤
 ** au fost citite :        12  inregistrari < san    >  din care:
                  12  adaugari(creeri);
                   0  stergeri(suprimari);
                   0  modificari(actualizari);
                   0  eronate
t: 46/ 176/ 130
d: 3/ 113
%        RUN FN:0
** socrate started **  v1.5    le: 17/11/87  a 18.18.56.72.
             1    d x1 d x2
             2    m x1=formal macheta dans zona 1
             3    faire s x1
             4     lire dans zona 1
             5      m z1=u m z1=ident de x1 si z1 = '¤¤¤¤' alors sortie fin
             6     si z1^='nac' alors refaire fin
             7     m caracter = cod-nac de x1 m y1=numde caracter
             8     si y1^=u alors
             9      si pas un nationalitate-cetatenie y1
            10       alors g un nationalitate-cetatenie y1 fin
            11       m den1 de un nationalitate-cetatenie y1    =
            12            den1-nac de x1
            13      fin
            14     refaire fin?
t: 171/ 322/ 151
d: 2/ 383
            15    nac6turc
            16    nac7italian
            17    nac5sirb
            18    nac4grec
            19    nac3german
            20    nac8alte nationalitati
            21    nac2maghiar
            22    nac1roman
            23    ¤¤¤¤
t: 32/ 146/ 114
d: 3/ 91
%        RUN FN:R
** socrate started **  v1.5    le: 17/11/87  a 18.19.08.80.
             1    d x1 d x2 d x3
             2    d x2 = un persoana
```

```
m x1 = formal pers dans buf 1
faire
  s  x1 m y1 = u m y2 = u
  m z1 = u
  lire dans buf 1
  m z1 = ident de x1
  si z1 = '¤¤¤¤'  /* test sfirsit fisier cartele */
    alors
  i (1) '*creere-actualizare <persoana>'
  i (+1) 'terminata' ecrire
    sortie  /* iesire din ciclu faire ...refaire...fin   */
  fin
 si z1 = 'p01'
  alors
  m z2 = u m z2 = matricol de x1
  m x2 = un persoana avec cod = z2;
  si pas x2
    alors
    g un persoana x3
    m y1 = numde x3
    m cod de x3  = matricol de x1
    m x2 = x3
  fin
   exec act-p01
   m z1 = u m z1 = sex de x1
   si z1 = '1'
    alors
     g un barbati de x0 = x2
    sinon
     si z1 = '2'
      alors
       g un femei de x0 = x2
   fin fin
  sinon
   si z1 = 'p02'
    alors
     exec act-p02
    sinon
     i (5) '*err.cod macheta eronat' ecrire
     refaire
   fin
   refaire
 fin
   refaire
 fin
 ?
t: 284/ 614/ 330
d: 4/ 545
 p011501121020125cojocaru aurelian 40007000111pcr 19800805xyz000001
 p011471250100101sabau    gheorghe 40007000111pcr 19700110xyz000001
zi ****erreur  valeur numerique erronee
 p011521001030101avram    vasile   40007000111er 19800110xyz000001
apartenenta ****erreur  valeur de liste erronee
an ****erreur  valeur numerique erronee
 p012500821400011popa     elena    40007000111pcr 19770101
 p012501003400012dolis    florica  40007000111odus
 p011570101400011negru    adrian   40007000112utc
 p021570101400011
scrieti programul pentru tratarea machetei<p02>;
   eventual modificati programulprincipal pentru a trata
si macheta<p03>;scrieti secvende program corespondenta.
?(mult succes)?
<marca>;  1570101400011 <nume>:negru
  ¤¤¤¤
*creere-actualizare <persoana> terminata
t: 128/ 548/ 420
d: 34/ 277
```

```
%        RUN FN:R
** socrate started **  v1.5    le: 17/11/87  a 18.19.27.58.
              i '***structura personalului***'
    m y1 = d tout persoana
    i (1 ) 'total personal :' i (+10 -10) y1 i (+1)
     ',din care :' ecrire
    m y1 = d tout tesa de x0
     i (20) '-tesa:' i (+10 -10) y1  i (+0) ';' ecrire
    m y1 = d tout muncitori de x0
     i (20) ' muncitori:' i (+10 -10) y1 i (+0) ';' ecrire
     m y1 = d tout barbati
      i (20) '-barbati:' i (+10 -10) y1 i (+0) ';' ecrire
    m y1= d tout femei
     i (20) '-femei:'   i (+10 -10) y1 i (+0) '.' ecrire
    ?
t: 195/ 452/ 257
d: 0/ 305
***structura personalului***
total personal :          6  ,din care :
                     -tesa:             0;
                     -muncitori:           0;
                     -barbati:           4;
                     -femei:          2.
t: 10/ 58/ 48
d: 2/ 5
%        RUN FN:R
** socrate started **  v1.5     le: 17/11/87  a 18.19.35.04.
    d x1 d x2
    m y1 = 1 m y2 = 0 m y3 = 0
     pour tout persoana x1 par cod;
      m y3=y3 + 1
      si y2=0
       alors
        exec pageskip
    i (1) 'registrul numerelor matricole'
        i (100) 'pagina:'
        i (+5 -5) y1
        ecrire
         i ' '
        linast120
        i (1) '* nr. * matricol' i (21) '*nume si prenume'
      i (82) '*' i (95) '*observatii' i (120) '*'
         ecrire
        i (1) '*crt. *' i (21) '*' i (82 15) '*data nasterii*'
        i (120) '*' ecrire
        linast120
       m y2=5
       m y1=y1 + 1
     fin
        i (1) '*'  i (6 -5) y3 i (7) '*'
        i (+0) cod i (21) '*'
      i (+0) nume i (+1) prenume i (82) '*'
        i (87 -4) an de data-nasterii  i (87) '*'
      i (93 -2)  luna de data-nasterii i (93) '*'
      i (96 -2)  zi de data-nasterii i (96) '*' i (120) '*'
       ecrire
      m y2=y2 + 1
      si y2 > 85
       alors
        linast120
       m y2=0
      fin
     linast120
   fin ?
t: 419/ 938/ 519
d: 10/ 461
```

```
registrul numerelor matricole                                                              pagina:      1

****************************************************************************************************
* nr. * matricol      *nume si prenume                                  *             *observatii     *
*crt. *               *                                                 *data nasterii*               *
****************************************************************************************************
*   1 *1570101400011*negru adrian                                       *1957*   1* 1*                *
****************************************************************************************************
*   2 *1521001030101*avram vasile                                       *1952*  10* 1*                *
****************************************************************************************************
*   3 *2501003400012*dolis florica                                      *1950*  10* 3*                *
****************************************************************************************************
*   4 *2500821400011*popa elena                                         *1950*   8*21*                *
****************************************************************************************************
*   5 *1501121020125*cojocaru aurelian                                  *1950*  11*21*                *
****************************************************************************************************
*   6 *1471250100101*sabau gheorghe                                     *1947*  12*  *                *
****************************************************************************************************

t: 5633/ 12466/ 6833
d: 1170/ 71889
%          RUN FN:M
** socrate started **  v1.5     le: 17/11/87  a 18.22.02 40.
             1     :lisall?
  c pageskip                          1
  m linast120                         1
  m linast80                          1
  m capnom                            2
            m linast80
  m lisnom                            4
            c pageskip
            m capnom
            m linast80
  m cartela                           1
  c act-p01                           4
  c act-p02                           2
  m cam-jud                           3
            m cartela
  m cam-loc                           4
            m cartela
  m cam-spe                           4
            m cartela
  m cam-pro                           3
            m cartela
  m cam-fun                           3
            m cartela
  m cam-san                           3
            m cartela
  m cam-rec                           3
            m cartela
  m cam-grm                           3
            m cartela
** fin liste **
%          LOGOUT

0071  personal   an = 1010   ph = 0001   date = 17/11/87-321

h.deb = 18h 16m 52s   h.fin = 18h 22m 11s   time = 00011164   code = 000

lgp = 00110   mem = 00038   lo = 000416   in = 00255   out = 00000

          EOJ
```

TESTAREA COERENTEI FIZICE A BAZEI DE DATE

```
.       * APEL STATISTICA TOTALA
.       * DETERMINARE COERENTA FIZICA BAZA DE DATE
.       XPROC *,DV:AD7,GN:1,VN:0,FN:'PERSONALSOU/GBASGEN'
.       XPROC PROCGEN,MOD                                                  1
.       MOD &MODUL:SOCGBAS&                                                2
.       MOD &SUPB:DV:AD0&                                                  3
.       ENDMOD                                                             4
.       FETCH LN:BIBIMTSO,GN:1,VN:1,FN:SOCGBAS,                    *       1
.                 DV:AD0                                                   2
.       ASSIGN S,DVT:AD,VS:AD1AD1                                          3
.       ASSIGN Q,DVT:AD,VS:AD1AD1                                          4
.       LABEL S,AM:OFL,FN:'BAZA DE BAZE'                                   5
.       LABEL Q,AM:OFL,FN:'MANEVRA SGBD'                                   6
.       * ASSIGN Z,DV:MT00                                                 7
.       * LABEL Z,FN:'JURNAL'                                              8
.       ASSIGN T,DV:AD7                                                    9
.       * APPROCHE FIN PROCEDURE CATALOGUEE
.       EST                         FIN DE LA PROCEDURE CATALOGUEE
.       OPTION CS'CF:50,DS:4096'                                           5
.       RUN TIME:999,NL:500000,AD:0,0,KEY:60                               6
socgbas  started
nb cadre fich: 50
%       SOC BN:BDDPERS,PN:AVRAM,AN:0001,PW:SOC                             7
%       SYSRUN FN:S,ST:ALL                                                 8
***  statistiques etablies sur la base : bddpers ***
        espace des fichiers
nb ss-pages occupees : 1512        sur : 57488       ( 2 %)
nb squatters : 61          sur : 1512        ( 4 %)
nb ss-pages dont le haut est vide : 366 soit : 2652 octets
nb ss-pages dont le bas  est vide : 55  soit : 933  octets
nb de chaines a  1 elements :  1390    soit :  0  changements de pages et  0
nb de chaines a  2 elements :  61      soit :  61  changements de pages et  0
pour l'exploration des chaines il y a eu :
 1390     fois  0  changements de page
 61       fois  1  changements de page
 1451     fois  0  changements de cylindre
***  statistiques etablies sur la base : bddpers ***
        espace des  dictionnaires
nb ss-pages occupees : 16          sur : 76352       ( 0 %)
nb squatters : 1          sur : 16           ( 6 %)
nb ss-pages dont le haut est vide : 0           soit : 0          octets
nb ss-pages dont le bas  est vide : 16          soit : 64         octets
nb de chaines a  1 elements : 14 soit :  0 changements de pages et  0
nb de chaines a  2 elements :  1  soit :  0 changements de pages et  0
pour l'exploration des chaines il y a eu :
 15       fois  0  changements de page
 15       fois 0  changements de cylindre
***  statistiques etablies sur la base : bddpers ***
        espace des  programmes
nb ss-pages occupees : 7          sur : 1193        ( 0 %)
nb squatters : 0           sur : 7           ( 0 %)
nb ss-pages dont le haut est vide : 4 soit : 4  octets
nb ss-pages dont le bas  est vide : 0 soit : 0  octets
nb de chaines a  1 elements :  7    soit :  0 changements de pages et  0
pour l'exploration des chaines il y a eu :
 7        fois 0  changements de page
 7        fois  0  changements de cylindre
fin prog utilitaire
%       LOGOUT                                                             9
0069  personal   an = 1010   ph = 0001   date = 17/11/87-321
h.deb = 18h 09m 47s   h.fin = 18h 16m 49s   time = 00014269   code = 000
lgp = 00110   mem = 00030   lo = 000088   in = 00029   out = 00000
.       * APPROCHE FIN PROCEDURE CATALOGUEE
.       EST                         FIN DE LA PROCEDURE CATALOGUEE
```

TESTAREA COERENTEI LOGICE

Exemplele care urmeaza in continuare arata modul de utilizare, pentru un caz particular, a programelor generalizate construite in acest scop.

```
.        * EXEMPLE DE TESTE COERENTA LOGICA COMPONENTE

.        XPROC PROCGEN,MOD
.        MOD &SUPB:DV:ADO&
.        ENDMOD

.        FETCH LN:BIBIMTSO,GN:1,VN:1,FN:SOCBTCH.              *
.              DV:ADO
.        ASSIGN S,DVT:AD,VS:AD1AD1
.        ASSIGN Q,DVT:AD,VS:AD1AD1
.        LABEL S,AM:OFL,FN:'BAZA DE BAZE'
.        LABEL Q,AM:OFL,FN:'MANEVRA SGBD'
.        * ASSIGN Z,DV:MT00
.        * LABEL Z,FN:'JURNAL'
.        ASSIGN T,DV:AD7
.        * APPROCHE FIN PROCEDURE CATALOGUEE
.        EST                           FIN DE LA PROCEDURE CATALOGUEE

.        OPTION CS'CF:50,DS:4096'
.        RUN TIME:999,NL:500000,AD:0,0,KEY:60

socbtch  started

nb cadre fich: 50

%        SOC BN:BDDPERS,PN:AVRAM,PW:SOC,AN:0001

%        RUN FN:0
** socrate started **  v1.5   le: 17/11/87  a 18.22.33.20.

              /* program pentru testare coerenta logica a      */
              /* caracteristicilor de tip:                      */
              /*        <anneau>          */
           m y1 = 0  m y2 = 0  m y3 = 0 /* anulare variabile */
           d x1  d x2                   /* de lucru<socrate> */
          pour tout localitati x1
            m y1 = d tout locn-pers de x1
            m y2 = y2 + y1
            fin
            pour tout persoana x2
             si existe pers-locn de data-nasterii de x2
               alors
                 m y3 = y3 + 1
             fin
           fin
           i (+1 )   '<nr.persoane grupate pe locali'
           i (+0 )   'tatea de nastere>:' i (+10 -10) y2
           i (+1 )   '<nr.pers.care refera o localit'
           i (+0 )   'ate>:' i (+10 -10) y3
           si y2 = y3
             alors
              i (+1) '??at..inc. ??'
           fin
           ecrire ?

<nr.persoane grupate pe localitatea de nastere>:          0

<nr.pers.care refera o localitate>:          0

%        RUN FN:0
```

```
** socrate started **  v1.5   le: 17/11/87  a 18.22.43.72.

         m yl = 0 m y2 = 0  /* coerenta<inverse> */
         m yl = d tout barbati
         pour tout barbati m y2 = y2 + 1 fin
         i yl i y2 si yl ^= y2 alors i '! att.inc.logica !!' fin ?

yl : 4

y2 : 4

%       RUN FN:0
** socrate started **  v1.5   le: 17/11/87  a 18.22.50.62.

         /*   <caracteristici cu cheie       */
         m yl = 0 m y2 = 0
         d xl  d x2
         pour tout persoana par cod:
          m yl = yl + 1
         fin
         pour tout persoana x2
          si existe cod
           alors
             m y2 = y2 + 1
          fin
         fin
         i ( 1 ) '<nr.coduri in dictionar>:'
         i (+10 -10) yl i (+1) ';' i (+1) '<nr.total coduri>:'
         i (+10 -10) y2
         si yl ^= y2
          alors
           i (+3) '!! atentie incoerenta logica !' fin ecrire
         ?

<nr.coduri in dictionar>:        6 ;

<nr.total coduri>:        6

%       LOGOUT

 0072  personal   an = 1010   ph = 0001   date = 17/11/87-321

 h.deb = 18h 22m 12s   h.fin = 18h 25m 39s   time = 00006490   code = 000

 lgp = 00110   mem = 00038   lo = 000096   in = 00076   out = 00000

          EOJ
```

ANEXE

ANEXA 1

IMAGINEA MEMORIEI VIRTUALE A BAZEI DE DATE <BDDPERS>
(rezumat)

adresse	identificateur	frer-per	fils	typ	cmax	origine	tail ptpr	org adrdic	nbmoy
00000100	fichier*********	00000000	0000013c	19	1	65536	26cd9349**	00000000**	00000000**
0000013c	raspuns*********	00000178	00000000	15	0	65536	001f8000**	00000000**	00000000**
00000178	rasp************	000001b4	00000000	15	0	65543	18108000**	00000000**	00000000**
000001b4	continuati******	0000022c	000001f0	16	4	65547	18038000**	00000000**	00040004**
000001f0	Bda Bnu Ay An								
0000022c	mnemonica*******	00000268	00000000	15	0	65548	00058000**	00000000**	00000000**
00000268	codnum**********	000002a4	00000001	10	0	65550	001e8000**	00000000**	3b9ac9fe**
000002a4	codalf**********	000002e0	00000000	15	0	65551	001f8000**	00000000**	00000000**
000002e0	caracter********	00000448	0000031c	16	64	65558	18078000**	00000000**	0004000f**
0000031c	1 2 3 4 5 6 7 8 9 a b c d e f								
00000358	00000394 000003d0 0000040c								
00000448	muncitori*******	000004c0	000006a0	3*	36864	65559	00000001**	10000481**	00000000**
00000484	persoanai 7 8 9 a b c d e f								
000004c0	tesa************	00000538	000006a0	3*	36864	66713	00000001**	10000481**	00000000**
000004fc	d persoanai 7 8 9 a b c d e f								
00000538	invgen**********	000005b0	000006a0	3*	36864	67867	00000001**	10000481**	00000000**
00000574	persoanai 7 8 9 a b c d e f								
000005b0	barbati*********	00000628	000006a0	3*	36864	69021	00000001**	10000481**	00000000**
000005ec	persoanai 7 8 9 a b c d e f								
00000628	femei***********	000006a0	000006a0	3*	36864	70175	00000001**	10000481**	00000000**
00000664	persoanai 7 8 9 a b c d e f								
000006a0	persoana********	00001bb8	000006dc	2*	36864	73635	00000043**	10000481**	00000000**
000006dc	persoana********	fffff960	00000718	19	1	0	00000043**	00000000**	00000000**
00000718	cod*************	00000754	00000000	15	73727	0	000e8000**	9000004e**	00000000**
00000754	nume************	00000790	00000000	15	73727	3	10118000**	840000de**	00000000**
00000790	prenume*********	000007cc	00000000	15	0	7	18118000**	00000000**	00000000**
000007cc	data-nasterii***	00000934	00000808	19	1	12	00000003**	00000000**	00000000**
00000808	an**************	00000844	0000076c	10	0	12	00088000**	00000000**	000000c8**
adresse	identificateur	frer-per	fils	typ	cmax	origine	tail ptpr	org adrdic	nbmoy
00000844	luna************	00000880	00000001	10	0	12	08048000**	00000000**	0000000b**
00000880	zi**************	000008bc	00000001	10	0	12	0c058000**	00000000**	0000001e**
000008bc	pers-locn*******	fffff834	00001ce4	7*	0	13	00000002**	00000026**	00001e88**
000008f8	■locn-perslocalitati								
00000934	cetatenie*******	000009ac	00000970	16	15	15	00048000**	00000000**	0004000f**

ANEXA 2

PROCEDURA LINKCOBO

```
BIBLIO   000.000   05/12/79   20H 24M 20S

NOM DU FICHIER       LINKCOBO          NUMERO DE MISE A JOUR  :   1
     01     .     SEG SGMAITRE
     02     .     ENTSG SGBD
     03     .     ENTSG SOPEN
     04     .     ENTSG SCLOSE
     05     .     ENTSG SSAVE
     06     .     ENTSG STERM
     07     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MTC010
     08     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MTC020
     09     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MTB050
     10     .     SEG SGINOUT
     11     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INB070
     12     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INB010
     13     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INB020
     14     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INB030
     15     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INB040
     16     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INB050
     17     .     MODIFY 0834T610A07F8                         V1,5-00 40-01/06
     18     .     MODIFY 0838I8639001C                         V1,5-00 40-02/06
     19     .     MODIFY 083CI3C030000                         V1,5-00 40-03/06
     20     .     MODIFY 0840T60000887                         V1,5-00 40-04/06
     21     .     MODIFY 0844T61360838                         V1,5-00 40-05/06
     22     .     MODIFY 0884I0000000D                         V1,5-00 40-06/06
     23     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INB060
     24     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INB080
     25     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INC010
     26     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1INC020
     27     .     SEG MONITEUR
     28     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,VN:1,GN:1,FN:V1MOC010
     29     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MOC020
     30     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MOC060
     31     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MOC100
     32     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MOC120
     33     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MOC300
     34     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MON021
     35     .     MODIFY 04E4T605A053C                         V1,5-00 23-01/05
     36     .     MODIFY 0520T01340528                         V1,5-00 23-02/05
     37     .     MODIFY 0524I052E0000                         V1,5-00 23-03/05
     38     .     MODIFY 0528T600A053C                         V1,5-00 23-04/05
     39     .     MODIFY 052CT6B2D053C                         V1,5-09 23-05/05
     40     .     MODIFY 0590T603805B8                         V1,5-00 24-01/09
     41     .     MODIFY 0594T6A2C053C                         V1,5-09 24-02/09
     42     .     COMPILE ASSIRIS                              V1,5-00 24-03/09
     43  CORRECT   CSECT                                       V1,5-00 24-04/09
     44             REF    TACHCOUR,LOADCOM,ERRPGM             V1,5-00 24-05/09
     45             LD4,11 TACHCOUR                            V1,5-00 24-06/09
     46             BAL,8  LOADCOM                             V1,5-09 24-07/09
     47             BRU    ERRPGM+X'1C'                        V1,5-00 24-08/09
     48             END                                        V1,5-00 24-09/09
     49     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MON030
     50     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MON061
     51     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MON070
     52     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MON190
     53     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MON250
     54     .     FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1MUL161
     55     .     MODIFY 0038T68390470                         V1,5-00 30-01/06
     56     .     MODIFY 0224I20110006T0C350230                V1,5-00 30-02/06
     57     .     MODIFY 0204I20110006T003502D0                V1,5-00 30-03/06
     58     .     MODIFY 0470I050A0010IA5210014                V1,5-00 30-04/06
     59     .     MODIFY 0478I05160010I251E0010                V1,5-00 30-05/06
     60     .     MODIFY 0480T08880000                         V1,5-00 30-06/06
```

(continuare) ANEXA 2

```
FETCHB LN:BIBRBNSO,DV:&D:AD0&,VN:1,GN:1,FN:V1MON162
FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1PAG010
FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1PAG020
FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1PAG030
FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1PAG040
FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1PAG051
FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1PAG060
SEG SGCONV
FETCHB LN:BIBRBNSO,DV:&D:AD0&,GN:1,VN:1,UN:255,FN:V1CVN010
MODIFY 0734I002A00F0                          V1.5-00 18-01/01
SEG SGBBFCT
FETCHB LN:BIBRBNSO,DV:&D:AD0&,VN:1,GN:1,UN:255,FN:V1BBF010
MODIFY 0088I10003000                          V1.5-00 33-06/07
MODIFY 00A0I10002FB8                          V1.5-00 04-01/13
MODIFY 00D0I10002FF4                          V1.5-00 04-02/13
MODIFY 0878I00350000                          V1.5-00 21-01/03
MODIFY 087CI00350000                          V1.5-00 21-02/03
MODIFY 0C08I500A012C                          V1.5-00 21-03/03
MODIFY 1524I00280000I30580144                 V1.5-00 33-01/07
MODIFY 152CI8639001CI688F0000                 V1.5-00 33-02/07
MODIFY 1534I80000004I80000008                 V1.5-00 33-03/07
MODIFY 153CI30000144I1038155C                 V1.5-00 33-04/07
MODIFY 1544I30580144                          V1.5-00 33-05/07
MODIFY 2354I10382FDCI01210401                 V1.5-00 04-03/13
MODIFY 2AE4I10382ABC                          V1.5-00 04-04/13
MODIFY 2D18I10383008                          V1.5-00 39-01/22
MODIFY 2FB8I012A0000I000A0014                 V1.5-00 04-05/13
MODIFY 2FC0I03960000I00270001                 V1.5-00 04-06/13
MODIFY 2FC8I020A0000I006B061F                 V1.5-00 04-07/13
MODIFY 2FD0I03E80004I01210401                 V1.5-00 04-08/13
MODIFY 2FD8I08B80000I52080080                 V1.5-00 04-09/13
MODIFY 2FE0I000A0004I012A0000                 V1.5-00 04-10/13
MODIFY 2FE8I006B061FI00680008                 V1.5-00 04-11/13
MODIFY 2FF0I10382358I03280030                 V1.5-00 04-12/13
MODIFY 2FF8I525A00F4I08B80000                 V1.5-00 04-13/13
MODIFY 3000I00280000I10381544                 V1.5-00 33-07/07
MODIFY 3008I000A0000                          V1.5-00 39-02/22
MODIFY 300CI18353018                          V1.5-00 39-03/22
MODIFY 3010I00120028                          V1.5-00 39-04/22
MODIFY 3014I10382D1C                          V1.5-00 39-05/22
MODIFY 3018I8639001C                          V1.5-00 39-06/22
MODIFY 301CI180F0000                          V1.5-00 39-07/22
MODIFY 3020I00000490                          V1.5-00 39-08/22
MODIFY 3024I30000000                          V1.5-00 39-09/22
MODIFY 3028IC1D5D6D2                          V1.5-00 39-10/22
MODIFY 302CID6D2C3C2                          V1.5-00 39-11/22
MODIFY 3030I8639001C                          V1.5-00 39-12/22
MODIFY 3034I180F0000                          V1.5-00 39-13/22
MODIFY 3038I00000200                          V1.5-00 39-14/22
MODIFY 303CI50000000                          V1.5-00 39-15/22
MODIFY 3040IC1D5D6D2                          V1.5-00 39-16/22
MODIFY 3044ID6C3C4D7                          V1.5-00 39-17/22
MODIFY 3048I585A0090                          V1.5-00 39-18/22
MODIFY 304CI8639001C                          V1.5-00 39-19/22
MODIFY 3050IB883000E                          V1.5-00 39-20/22
MODIFY 3054I580A0090                          V1.5-00 39-21/22
MODIFY 3058I10382D44                          V1.5-00 39-22/22
SEG MAJIS
SEG INTRPRE
SEG SGTRACE
TREE &TREE&%00(SGMAITRE(MONITEUR(MAJIS,INTRPRE,
              SGCONV(SGBBFCT,SGINOUT(SGTRACE)))))

% DISPLAY OL:A.OF:GENBDB
```

ANEXA 3

PROCEDURĂ GENERALIZATĂ PENTRU REFACEREA LEGĂTURILOR DE TIP ⟨*referire cu inel*⟩

Parametrii formali utilizați în descrierea procedurilor de extragere/refacere a caracteristicilor de tip ⟨*inel*⟩ cu semnificația :

— ⟨*entiann*⟩ : numele entității care conține caracteristica ⟨*inel*⟩ ;
— ⟨*inel*⟩ : numele caracteristicii de tip ⟨*inel*⟩ ;
— ⟨*entiref*⟩ : numele entității care conține caracteristica de tip ⟨referire⟩ ;
— ⟨*referire*⟩ : nume caracteristica de tip ⟨*referire*⟩ ;
— ⟨*codind*⟩ : codul realizării care conține caracteristica de tip ⟨*inel*⟩ ;
— ⟨*codann*⟩ : caracteristica cheie discriminantă din ⟨*entiann*⟩ ;
— ⟨*cod ref*⟩ : caracteristica cheie discriminativă din ⟨*entiref*⟩ ;
— ⟨*tipcodann*⟩ : tipul de cod ⟨*codann*⟩ ;
— ⟨*tipcodref*⟩ : tipul de cod ⟨*codref*⟩ ;
— ⟨*tipcod xxx*⟩ : : = ALF/NUM/VID
① ALF : alfanumeric ;
② NUM : numeric ;
③ VID : număr de realizare.

În procedurile prezentate utilizatorul va înlocui parametrii formali, acolo unde apar în text, cu valorile atribuite la apel. În funcție de valoarea parametrului ⟨*tip cod* xxx⟩ va alege secvențele cu același număr din procedură.

— ⟨*tiprel*⟩ : tipul de prelucrare ales definit ca punct de intrare :
 — EXTRAG : recuperare valori ⟨*referire*⟩ ;
 — REFAC : refacere legături
— ⟨*identi*⟩ : numele fișierului pe/de pe care se execută extragerea/refacerea.

Indiferent care este tipul de prelucrare ales, secvențele SOCRATE adaptate la cerințele utilizatorului vor fi precedate de liniile de comandă cu următoarea structură de principiu :

• ⟨*tiprel*⟩ * ⟨*entiref*⟩
• XPROC PROCGEN
• OPTION CS'CF : 90, OS : 4096'
• ⟨*tiprel*⟩ RUN TIME : 999,NL : 500000,AD : 0,0
% SOC BN : ⟨*nume baza*⟩, PN : ⟨*nume uti*⟩, AN : ⟨*cont uti*⟩, PW : *parola*
% ATTACH MTOO
% LABEL FN :' ⟨*identi*⟩'
% ASSIGN DV : MTOO
% FILE RCS : 60, BFS : 2041, RCF : FIX, PRM : {OUT pentru EXTRAG / INP pentru REFAC}
% RUN FN : R

codul sursă al programului adaptat de utilizator conform specificațiilor pentru fiecare punct de intrare.

• SKIP ERR, COND : (127, LE, ⟨TIPREL⟩)

Punctul de intrare ⟨ERR⟩ are structura :

• SKIP SF
• ERR *
• ** INCIDENT FATAL ÎN PROCEDURA DE PRELUCRARE !
• ** ANALIZAȚI EROAREA ȘI RELUAȚI CONFORM INDICAȚIILOR !
SF *

```
1) Punctul de intrare <EXTRAG>
   D X1 D X2 D X3
   M X1 = FORMAL REFAC - LEGATUI DANS BUF O S X1
   FAIRE M Z1 = U M Y1 = U
   dacă <tipcodann> = 'ALF'
      atunci
     ┌─────────────────────────────────────────────
     │ M Z1 = '<codind>'
   1 │ M CODANN-ALF DE X1 = Z1
     │ M X2 = UN <entiann> AVEC <codann> = Z1 ;
     └─────────────────────────────────────────────
altfel
   dacă <tipcodann> = 'NUM'
      atunci
     ┌─────────────────────────────────────────────
     │ M Y1 = <codind>
   2 │ M CODANN-NUM DE X1 = Y1
     │ M X2 = UN <entiann> AVEC <codann> = Y1 ;
     └─────────────────────────────────────────────
      altfel
            dacă <tipcodann> = 'VID'
                 atunci
     ┌──────────────────────────────
     │ M Y1 ; <codind>
   3 │ M CODANN-NUM DE X1 = Y1
     │ M X2 = UN <entiann> Y1
     └──────────────────────────────
            sfd ;
       sfd ;
   sfd ;
```

/* Secvenţa precedentă permite utilizatorului să aleagă, în funcţie de ⟨*tipcodann*⟩. modul de stabilire a realizării care conţine capul de lanţ. Indiferent care este numărul secvenţei alese este permisă introducerea unei ⟨*calificări*⟩ a entităţii ⟨*entiann*⟩ conformă cu modelul structural al datelor.
*/

```
   SI PAS X2
      ALORS
         I (1) '** ERR.FF.NU EXISTĂ O REALIZARE'
         I (+1) '<IDENTI>'
         ECRIRE M Y5 = -65535 I (Y5) " ECRIRE
         SORTIE
   FIN
   M Y5 = 0
   POUR TOUT <entiref> X3
      AYANT <referire> = X2 ;
      M Y5 = Y5 + 1
```

```
    dacă ⟨tipcod ref⟩ = 'ALF'
    | atunci
1   | | M CODREF-ALF DE X1=⟨codref⟩ DE X3 |
    | altfel
    | dacă ⟨tipcod ref⟩ = 'NUM'
    | | atunci
    | | | M CODREF-NUM DE X1 = ⟨codref⟩ DE X3 |
    |2| altfel
    | | dacă ⟨tipcodref⟩ = 'VID'
    | | | atunci
    | | | | M Y6 = NUMDE X3
    | |3| | M CODREF-NUM DE X1 = Y6 |
    | | sfd ;
    | sfd ;
    sfd ;
    M ⟨referire⟩ = U /* ANULEAZĂ REFERIRE
    ECRIRE BUF O DANS BANDA O /* SCRIE ARTICOLUL */
FIN /* POUR */
M Z1 = '•OEF'                     /* MARCHEAZĂ          */
M CODANN-ALF DE X1 = Z1           /* SFÎRȘIT LOGIC      */
ECRIRE BUF O DANS BANDA O         /*FIȘIER              */
I (1) '**** AU FOST RECUPERATE :' I (+10 −10) Y5
I (+1) 'REALIZĂRI DIN' I(+1) '⟨entiref⟩'
I (+1) 'CARE REFERĂ' ECRIRE
I (+1) 'REALIZAREA :' I (+1) ⟨codind⟩ I(+1) 'DIN'
I (+1) '⟨entiann⟩ ;' ECRIRE
SE UN ⟨inel⟩ DE X2
SI EXISTE ⟨inel⟩ DE X2
  ALORS
    I (1) '**** ATENȚIE ! ANULAȚI' I (+1) '⟨inel⟩'
    I (+1) 'DIN' I (+1) '⟨entiann⟩' I (+1)'?' ECRIRE
    I (5) 'DUPĂ ANULARE RELUAȚI PROCEDURA'
    I (+1) 'DIN PUNCTUL DE INTRARE ⟨REFAC⟩.' ECRIRE
    M Y5 = −65536 /* poziționarea codului de retur al */
    I (Y5) '' ECRIRE /* fazei la o valoare mai mare ca 127 */
  FIN
FIN /* FAIRE */ D X1 D X2 D X3
?
% LOGOUT
• * SFÎRȘIT
2) punctul de intrare ⟨REFAC⟩
  D X1 D X2 D X3
  M Y4 = 0 M Y5 = 0
```

```
M X1 = FORMAL REFAC – LEGĂTURI DANS ZONA O
FAIRE
   S X1 /* anulare zona de citire a articolelor */
   LIRE ARTICOL O DANS ZONA O
   M Z2 = U M Z2 = CODANN-ALF DE X1
   SI Z2 = '.EOF'
     ALORS
       I(1) '**** AU FOST REFĂCUTE ;'
       I(+10 –10) Y5 I(+1) 'LEGĂTURI' I(+1) '⟨referire⟩'
       I(+1) 'DIN' I(+10 –10) Y4 I (+1) 'CITIRI DIN BANDA.'
       ECRIRE
       SORTIE
   FIN
   M Y4 = Y4 + 1
   M Z1 = U M Y1 = U
dacă ⟨tipcodann⟩ = 'ALF'
 |   atunci
1|   | M Z1 = CODANN-ALF DE X1                  |
 |   | M X2 = UN ⟨entiann⟩ AVEC ⟨codann⟩ = Z1 ; |
 |   altfel
 |     dacă ⟨tipcodann⟩ = 'NUM'
 |     |  atunci
 |   2 |  | M Y1 = CODANN-NUM DE X1                   |
 |     |  | M X2 = UN ⟨entiann⟩ AVEC ⟨codann⟩ = Y1 ; |
 |     |  altfel
 |     |  dacă ⟨tipcodann⟩ = 'VID'
 |     |  |  atunci
 |   3 |  |  | M Y1 = CODANN-NUM DE X1 |
 |     |  |  | M X2 = UN entian Y1     |
 |     |  |sfd ;
 |     sfd ;
sfd ;
SI PAS X2
  ALORS
    I (1) 'xxERR, FF, NU EXISTĂ...' I (+1)' ⟨entiann⟩'
    I (+1) 'xxxPRELUCRARE ABANDONATĂ xxx' ECRIRE
    M Y1 = –65536 I (Y1) '' ECRIRE
    SORTIE /* inutil ? */
FIN
M Z2 = U M Y2 = U
```

```
dacă ⟨tipcodref⟩ = 'ALF'
 | alors
 |  ---------------------------------------
 | | M Z2 = CODREF-ALF DE X1               |
1| | M X3 = UN ⟨entiref⟩ AVEC ⟨codref⟩ = Z2 ; |
 |  ---------------------------------------
 | altfel
 |   dacă ⟨tipcodref⟩ = 'NUM'
 |   | atunci
 |   |  ---------------------------------------
 |   | | M Y2 = CODREF-NUM DE X1               |
 |  2| | M Z2 = Y2                             |
 |   | | M X3 = UN ⟨entiref⟩ AVEC ⟨codref⟩ = Y2 ; |
 |   |  ---------------------------------------
 |   | altfel
 |   |   dacă ⟨tipcodref⟩ = 'VID'
 |   |   | atunci
 |   |   |  --------------------------
 |   |   | | M Y2 = CODREF-NUM DE X1  |
 |   |  3| | M Z2 = Y2                |
 |   |   | | M X3 = UN ⟨entiref⟩ Y2   |
 |   |   |  --------------------------
 |   |   sfd ;
 |   sfd ;
sfd ;

SI PAS X3
   ALORS
   I (1) '**ERR.FF.NU EXISTĂ...' I (+1) '⟨entiref⟩'
   I (+1) Z2 ECRIRE
   REFAIRE
 FIN
 M ⟨referire⟩ DE X3 = U
 M ⟨referire⟩ DE X3 = X2
 M Y5 = Y5 + 1
 REFAIRE
FIN /* REFAIRE */
M Y6 = D TOUT ⟨inel⟩ DE X2
SI Y6 ¬= Y5
 ALORS
 I (1) '****GRAV****NUMĂR LEGĂTURI' I (+1)' PE INEL :'
 I (+10 -10) Y6 I (+1) 'DIFERIT DE NUMĂR REFACERI :'
 I (+10 -10) Y5 ECRIRE
 M Y5 = -65535 I (Y5) '' ECRIRE
FIN
D X1 D X2 D X3
?
% LOGOUT
• * SFÎRŞIT
```

ANEXA 4

1. Principiul de funcţionare al unui program de tip „CAM"

CAM0. start ;
CAM1. anulare variabile de lucru ;
CAM2. citeşte o înregistrare ;
CAM3. DACĂ ⟨*ident*⟩ = '$$$'
    ATUNCI scrie mesaj terminare prelucrare ; mergi la CAM5 ;
    ALTFEL
        DACĂ există o realizare cu codul ⟨*cod*⟩
            ATUNCI
                DACĂ ⟨*den*1⟩ = vid
                    ATUNCI şterge realizarea ;
                    ALTFEL modifică denumirea prin ⟨*den*1⟩, ⟨*den*2⟩,... ⟨*deni*⟩ ;
            ALTFEL crează o realizare cu codul ⟨*cod*⟩ şi denumire ⟨den_1⟩, ⟨den_2⟩, . . ., ⟨den_i⟩ ;
    DACĂ ⟨ref_1⟩ (⟨ref_2⟩, . . ., ⟨ref_i⟩) este prezent
        ATUNCI
            DACĂ există o realizare a entităţii referite
                ATUNCI realizează legătura ;
                ALTFEL scrie mesaj de eroare ;
        ALTFEL execută pasul următor ;
CAM.4 reluare de la CAM2 ;
CAM.5 sfîrşit.

2. Program pentru citare parţială „⟨*indicativ entitate*⟩"

; DEFMAC ⟨*indicativ entitate*⟩ : : EXP

$$\text{UN } \langle \textit{nume entitate} \rangle \begin{Bmatrix} \text{AVEC} \\ \text{AYANT} \end{Bmatrix} \begin{Bmatrix} \text{COD} \\ \langle \text{RCODE} \rangle \end{Bmatrix} = \begin{Bmatrix} :1 \\ ':1:' \end{Bmatrix} ;\ \text{:FDEF ?}$$

Notă : acest tip de citaţie este cerut în conversaţional la încărcarea caracteristicilor de tip referinţă. Numele programului de citaţie este indicat de numele caracteristicii de tip referinţă.

ANEXA 5

EXEMPLU DE PROTOTIP DE GENERARE

```
.       SYSRUN BIBLIO
biblio   started
biblio   000.000   17/11/87   17h 43m 25s
% OPNLIB R,LN:Z%PROC,GN:1,VN:0,DV:ADO,FT:SOU
% DISPLAY OL:R,OF:PROTCAM
n o m  d e  l a  b i b l i o t h e q u e  :          z%proc
        n u m e r o  d e  v e r s i o n  :              00
  n u m e r o  d e  g e n e r a t i o n  :            0001
                              f o r m a t  :           sou
nom du fichier  :   protcam   numero de mise a jour  :   1
    01              mgo         100,tipenti,eq,'num'
    02              mgo         100,tipenti,eq,'alf'
    03              mstring     tipenti='num'
    04    100       mnop
    05              mgo         200,tipref,eq,'num'
    06              mgo         200,tipref,eq,'num'
    07              mstring     tipref='alf'
    08    200       mnop
    09              mstring   a=''''
    10              mstring   pp='.'
    11              mstring   p='%'
    12              mstring   sp='
    13              mstring   d=':'
    14              mindex    ind=&ml(enti)
    15              mindex    k=0
    16          mstring   sl='/*','*/'
    17              mstring   bdd='bn:coral,pw:cc,pn:avram,an:1'
    18              mindex      i=1
    19              mindex      n=&ml(enti)
    20    &p&sp(1,8) soc &bdd   <sterge>
    21              mpar      mcom=0,mlab=71
    22              mark        5000
    23    500       mnop
    24              mgo         6000,i,gt,n
    25              mstring     work='&enti(i)'
    26              mindex      l=&mc(work)
    27              mindex    j=0
    28    700       mnop
    29              mindex      j=j+1
    30              mgo         720,j,eq,1
    31              mgo         730,work(j,1),eq,'-'
    32              mgo         700
    33    720       mnop
    34              mstring    cnomac='&work(1,3)'
    35              mgo         740
    36    730       mnop
    37              mstring    cnomac='&work(1,2)&work(j+1,1)'
    38    740       mnop
    39              mstring    cnomenti='&work'
    40              mindex      i=i+1
    41              mindex      j=&enti(i)
    42              mindex      k=j
    43              mgo         790,j,eq,1
    44              mgo         790,j,eq,3
    45              mgo         760,j,eq,2
    46              mgo         760,j,eq,4
  biblio   000.000   17/11/87   17h 43m 31s
    47              mgo         6500
    48    760       mnop
    49              mindex      i=i+1
    50              mstring     entiref='&enti(i)'
    51              mindex      j=0
    52              mindex      l=&mc(entiref)
```

```
770     mnop
        mindex      j=j+1
        mgo         780,j,eq,1
        mgo         775,entiref(j,1),eq,'-'
        mgo         770
780     mnop
      . mstring     cnomref='&entiref(1,3)'
        mgo         785
6000    mnop
        mgo         7000
775     mnop
        mstring     cnomref='&entiref(1,3)&entiref(j+1,1)'
785     mnop
790   mnop
        mindex      i=i+1
 mpage
&p&sp(1,8) run fn:m
&d.supexp  cam-&cnomac  ?
&p&sp(1,8) run fn:m
 &d.defmac
  cam-&cnomac
 &d.exp
 d  x1  d  x2  d  x3  d x4    m y6 = 0   m y7 = 0
 m y1 = 0   m y2 = 0  m y3 = 0  m y4 = 0  m y5 = 0
 m x1 = formal cartela  dans  buf  0
 faire
  lire dans buf 0
  m z1 = ident        de        x1
  si z1 = &a.¤¤¤¤&a          alors  sortie  fin
  m  y4 = y4 + 1
  si z1 ^=  &a&cnomac&a
    alors
     m y7 = y7 + 1
     i &a.?? ident ?? *eronat* &a
     refaire
  fin
  m z5 = u   m y1 = u
    mgo          300,tipenti,eq,'num'
  m z5 = cod-&cnomac  de  x1
  m x2 = un &cnomenti avec cod = z5;
    mgo          310
biblio   000.000   17/11/87   17h 43m 32s
300     mnop
  m y1 = cod-&cnomac    de    x1
  m x2 = un  &cnomenti  avec cod = y1 ;
310     mnop
 si existe x2
    alors
    mgo  412,k,le,2
    mgo   434
6500    mnop
        mgo          7500
412  mnop
     m z2 = den1-&cnomac                    de x1
     si z2 = &a  &a
       alors
       m den1      de  x2    = u
       m cod       de  x2    = u
       s x2
       m y2 = y2 + 1    &sl(1) stergeri  &sl(2)
      sinon
       m den1      de  x2    = den1-&cnomac     de  x1
       m y6 = y6 + 1    &sl(1) modificari(actualizari) &sl(2)
     fin
 sinon
   g  un  &cnomenti            x3
   m  cod            de  x3    = cod-&cnomac      de  x1
```

```
      m den1          de  x3   = den1-&cnomac   de x1
      m  y3 = y3 + 1  &sl(1)  adaugari(creeri)  &sl(2)
    mgo  455
  7000      mnop
            mgo         9000
  434    mnop
       m z2 = den1-&cnomac               de  x1
       si z2 = &a  &a
         alors
           m den1     de  x2    = u
           m cod      de  x2    = u
           m den2     de  x2    = u
           m y2 = y2 + 1     &sl(1) stergeri &sl(2)
         sinon
           m den1     de  x2    = den1-&cnomac          de  x1
           m den2     de  x2    = den2-&cnomac          de  x1
           m y6 = y6 + 1    &sl(1) modificari(actualizari) &sl(2)
       fin
    sinon
      g  un  &cnomenti          x3
      m  cod          de  x3    = cod-&cnomac           de  x1
      m  den1         de  x3    = den1-&cnomac          de  x1
biblio    000.000   17/11/87   17h 43m 32s
      m  den2         de  x3    = den2-&cnomac          de  x1
      m y3 = y3 + 1         &sl(1)  adaugari(creeri)  &sl(2)
  455   mnop
    fin
            mgo           460,k,eq,1
            mgo           460,k,eq,3
            mgo           400,tipenti,eq,'num'
    m  x2 = un &cnomenti avec cod = z5;
            mgo           410
  7500      mnop
            mgo           600
  400       mnop
    m x2 = un &cnomenti        avec  cod = y1 ;
  410       mnop
            mgo           480,tipref,eq,'num'
    m z5 = ref1-&cnomac de x1
    m x4 = un &entiref avec cod = z5;
            mgo           450
  480       mnop
    m y5 = ref1-&cnomac  de x1
    m x4 = un   &entiref      avec  cod   = y5 ;
  450       mnop
    m &cnomac-&cnomref  de  x2  =  x4
  460       mnop
    refaire
    fin
    i (  1  ) &a ** au fost citite : &a
    i (+10 -10)  y4
    i (+1     )         &a inregistrari < &cnomac   > &a
    i (+0     )         &a din care: &a
    ecrire
    i (+20 -10)  y3
    i (+1     )         &a adaugari(creeri); &a
    ecrire
    i (+20 -10)  y2
    i (+1     )         &a stergeri(suprimari); &a
    ecrire
    i (+20 -10) y6
    i (+1     )         &a modificari(actualizari); &a
    ecrire
    i (+20 -10) y7
    i (+1     )         &a eronate  &a
    ecrire
  &d.fdef  ?
```

```
  185          mgo  500
  186     5000   mnop
  187     600    mnop
biblio    000.000    17/11/87    17h 43m 33s
  188     &pp&sp(1,8) ** ?? parametrii generare eronati ?? abandon ??
  189            mgo           999
  190     9000        mnop
  191     &p&sp(1,8) logout
  192     &pp&sp(1,8) ** o!k!<macro>
  193     99     mnop
  194     999  mnop
  195            mend

% DISPLAY OL:R,OF:CAM
biblio    000.000    17/11/87    17h 43m 42s

n o m   d e   l a   b i b l i o t h e q u e  :          z%proc
        n u m e r o   d e   v e r s i o n  :            00
   n u m e r o   d e   g e n e r a t i o n  :           0001
                                 f o r m a t  :         sou

biblio    000.000    17/11/87    17h 43m 42s

nom du fichier  :    cam              numero de mise a jour   :       1
    01    .         compile magiris,dv:&dv:ad0&,gn:&gn:1&,vn:&vn:0&,             *
    02    .                   ln:&ln:z%proc&,fn:&fn:camident&,nlg,obl
    03              mnote     <parametrii:  mod 3,3 >
    04              mfetch dv:&dvp:ad0&,ln:&lnp:z%proc&,gn:&gnp:1&,              *
    05                        vn:&vnp:0&,fn:&fnp:protcam&,un:&unp:255&
    06             mend
% DISPLAY OL:R,OF:PROCGEN

biblio    000.000    17/11/87    17h 43m 49s

n o m   d e   l a   b i b l i o t h e q u e  :          z%proc
        n u m e r o   d e   v e r s i o n  :            00
   n u m e r o   d e   g e n e r a t i o n  :           0001
                                 f o r m a t  :         sou
biblio    000.000    17/11/87    17h 43m 49s

nom du fichier  :    procgen          numero de mise a jour   :       1
    01    .         fetch ln:bibimtso,gn:&gn:1&,vn:&vn:1&,fn:&modul:socbtch&;    *
    02    .                   &supb:dvt:ad,vs:adlad1&
    03    .         assign s,&sups:dvt:ad,vs:adlad1&
    04    .         assign q,&sups:dvt:ad,vs:adlad1&
    05    .         label s,am:ofl,fn:'&fns:baza de baze&'
    06    .         label q,am:ofl,fn:'&fnq:manevra sgbd&'
    07    .         &jurnal:* &assign z,&supj:dv:mt00&
    08    .         &jurnal:* &label z,fn:'&fnj:jurnal&'
    09    .         assign t,dv:&supbd:ad7&

% ENDLIB


 display 17/11/87  18.01 nom du fichier : gbasgen                         3

   1:       .          xproc procgen,mod
   2:       .          mod &modul:socgbas&
   3:       .          mod &supb:dv:ad0&
   4:       .          endmod
   5:       .          option cs'cf:50,ds:4096'
   6:       .          run time:999,nl:500000,ad:0,0,key:60
   7:      &%&            soc bn:&bn:bddpers&,pn:&pn:avram&,an:&an:0001&,pw:&pw:soc&
   8:      &%&            sysrun fn:&functia:s&,st:&st:all&&bs&
   9:      &%&             logout
```

ANEXA 6. UN EXEMPLU DE STATISTICĂ PE BAZĂ DE DATE ADMITBD (BDDPERS REORGANIZATĂ PRIN SALVARE-RESTAURARE SOCRATE (partea I)

```
NB CADRE FICH: 50
%       SOC BN:ADMITBD,PN:AVRAM,PW:SOC,AN:2499
%       SYSRUN FN:S,ST:ALL

***  STATISTIQUES ETABLIES SUR LA BASE : ADMITBD ***

       ESPACE DES  FICHIERS
NB SS-PAGES OCCUPEES : 70802          SUR : 95968       ( 73%)
NB SQUATTERS : 16375        SUR : 70802        ( 23%)
NB SS-PAGES DONT LE HAUT EST VIDE : 37563          SOIT : 314827
NB SS-PAGES DONT LE BAS  EST VIDE : 9630           SOIT : 170264
NB DE CHAINES A  1  ELEMENTS :  44033          SOIT :  0
NB DE CHAINES A  2  ELEMENTS :  6717           SOIT :  6530
NB DE CHAINES A  3  ELEMENTS :  2083           SOIT :  3933
NB DE CHAINES A  4  ELEMENTS :  1052           SOIT :  2983
NB DE CHAINES A  5  ELEMENTS :  409            SOIT :  1557
NB DE CHAINES A  6  ELEMENTS :  101            SOIT :  481
NB DE CHAINES A  7  ELEMENTS :  29             SOIT :  166
NB DE CHAINES A  8  ELEMENTS :  3              SOIT :  17

POUR L'EXPLORATION DES CHAINES IL Y A EU :
 44220        FOIS  0  CHANGEMENTS DE PAGE
 6769         FOIS  1  CHANGEMENTS DE PAGE
 2016         FOIS  2  CHANGEMENTS DE PAGE
 954          FOIS  3  CHANGEMENTS DE PAGE
 359          FOIS  4  CHANGEMENTS DE PAGE
 86           FOIS  5  CHANGEMENTS DE PAGE
 23           FOIS  6  CHANGEMENTS DE PAGE

 53829        FOIS  0  CHANGEMENTS DE CYLINDRE
 569          FOIS  1  CHANGEMENTS DE CYLINDRE
 7            FOIS  2  CHANGEMENTS DE CYLINDRE
 21           FOIS  3  CHANGEMENTS DE CYLINDRE
 1            FOIS  4  CHANGEMENTS DE CYLINDRE


***  STATISTIQUES ETABLIES SUR LA BASE : ADMITBD ***

       ESPACE DES  DICTIONNAIRES
NB SS-PAGES OCCUPEES : 9355          SUR : 76352       ( 12%)
NB SQUATTERS : 712          SUR : 9355        ( 7 %)
NB SS-PAGES DONT LE HAUT EST VIDE : 9213          SOIT : 9213
NB SS-PAGES DONT LE BAS  EST VIDE : 142           SOIT : 568
NB DE CHAINES A  1  ELEMENTS :  7958           SOIT :  0
NB DE CHAINES A  2  ELEMENTS :  662            SOIT :  594
NB DE CHAINES A  3  ELEMENTS :  21             SOIT :  28
NB DE CHAINES A  4  ELEMENTS :  1              SOIT :  0
NB DE CHAINES A  6  ELEMENTS :  1              SOIT :  0

POUR L'EXPLORATION DES CHAINES IL Y A EU :
 8034         FOIS  0  CHANGEMENTS DE PAGE
 596          FOIS  1  CHANGEMENTS DE PAGE
 13           FOIS  2  CHANGEMENTS DE PAGE

 8643         FOIS  0  CHANGEMENTS DE CYLINDRE

***  STATISTIQUES ETABLIES SUR LA BASE : ADMITBD ***

       ESPACE DES  PROGRAMMES
NB SS-PAGES OCCUPEES : 9            SUR : 599        ( 1 %)
NB SQUATTERS : 0            SUR : 9           ( 0 %)
NB SS-PAGES DONT LE HAUT EST VIDE : 8             SOIT : 8
NB SS-PAGES DONT LE BAS  EST VIDE : 0             SOIT : 0
NB DE CHAINES A  1  ELEMENTS :  9             SOIT :  0

POUR L'EXPLORATION DES CHAINES IL Y A EU :
 9            FOIS  0  CHANGEMENTS DE PAGE

 9            FOIS  0  CHANGEMENTS DE CYLINDRE

FIN PROG UTILITAIRE
%       LOGOUT

.SALVB      JOB SALVBN95,AN:EE71,PN:AVRAM

                                        PROGRAMME DE MAINTENANCE

% SAVE FLE
% OLDFLE DV:AD1,FN:'ADMITBD'
% NEWFLE DV:MT1,FN:'ADMITBDAD0012'

% END
```

(partea a II-a)

```
OCTETS
OCTETS
CHANGEMENTS DE PAGES ET  C      CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  416    CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  46     CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  123    CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  27     CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  33     CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  5      CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  0      CHANGEMENTS DE CYLINDRE
```

```
OCTETS
OCTETS
CHANGEMENTS DE PAGES ET  0      CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  0      CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  0      CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  0      CHANGEMENTS DE CYLINDRE
CHANGEMENTS DE PAGES ET  0      CHANGEMENTS DE CYLINDRE
```

```
OCTETS
OCTETS
CHANGEMENTS DE PAGES ET  0      CHANGEMENTS DE CYLINDRE
```

```
          18

FICHIERS VERSION 15                  03/09/85  17H 47M

          4
          5
          6

          7
```

ANEXA 7. UN EXEMPLU DE INCOERENȚĂ FIZICĂ SEMNALATĂ LA EXECUȚIA UNUI STATISTIC (partea I)

```
NB CADRE FICH: 50
%       SOC BN:ADMITBD,PN:AVRAM,PW:SCC,AN:2499
%       SYSRUN FN:S,ST:ALL

*****   DUMP DE LA BASE : ADMITBD  *****        La intilnirea une

        ESPACE DES  FICHIERS                   Subspatiul FICHIE

*** TETE DE CHAINE ***                         Cap de lant

PAGE NUMERO :   781       ****

SS-PAGE  0  AV   1888537280      PAGE SUIVANTE [805]     SS-PAGE SU
      02E7F140  00000001  08430070  017E4040          088A007D


*** SQUATTER ERRONE ***

PAGE NUMERO :  [805]      ****                   Ar fi trebuit sa

SS-PAGE [28] AV   1888537280      PAGE SUIVANTE [0]       SS-PAGE SU
*** ERREUR *** SS-PAGE VIDE                    Subpagina este vida

*** TETE DE CHAINE ***

PAGE NUMERO :   781       ****

SS-PAGE  1  AV   1888537287      PAGE SUIVANTE  805      SS-PAGE SUI
      0903C1E3  05C7D6D9  C9C10000  0C9C007D          0401E5C5


*** SQUATTER ERRONE ***

PAGE NUMERO :  805        ****

SS-PAGE  27 AV   1888537287      PAGE SUIVANTE  0        SS-PAGE SUI
*** ERREUR *** SS-PAGE VIDE

*** TETE DE CHAINE ***

PAGE NUMERO :  781        ****

SS-PAGE  2  AV   1888537294      PAGE SUIVANTE  805      SS-PAGE SUI
      03C3D6C4  0843007D  017E4040  0C81007D          02E9F15E


*** SQUATTER ERRONE ***

PAGE NUMERO :  805        ****

SS-PAGE  25 AV   1888537294      PAGE SUIVANTE  0        SS-PAGE SUI
*** ERREUR *** SS-PAGE VIDE

*** TETE DE CHAINE ***

PAGE NUMERO :  781        ****

SS-PAGE  3  AV   1888537301      PAGE SUIVANTE  805      SS-PAGE SUI
      015E4040  081F007D  02E20940  0801007D          03D7C1E2


*** SQUATTER ERRONE ***

PAGE NUMERO :  805        ****

SS-PAGE  24 AV   1888537301      PAGE SUIVANTE  0        SS-PAGE SUI
*** ERREUR *** SS-PAGE VIDE
```

(partea a II-a)

11
17

i erori se declanșează automat DUMP pe zona care o conține

re

IVANTE 28
02E40540 1089007D° *BX1 AHC 'A= HI 'BUN EI ' *

punctează pe pagina 781

IVANTE 0 → *acest punctator ar fi corect deoarece în pagina 781 subpagina 0 este cap de lanț.*

IVANTE 27
C3404040 0889007D *ICATEGORIA <* 'DAVEC HI ' *

IVANTE 0

VANTE 25
00000001 0835007D *CCODHC 'A= <A 'BZ1; AH5 ' *

VANTE 0

VANTE 24
0080007D 02E7F140 *A; H= 'BSI HA 'CPAS< 'BX1 *

VANTE 0

ANEXA 7
(continuare partea I)

```
*** TETE DE CHAINE ***

PAGE NUMERO :  781    ****

SS-PAGE  21 AV  1888537427      PAGE SUIVANTE  855      SS-PAGE SUI

         10890070  0903C1E3  C5C7D6D9  09010000         009C0070

*** SQUATTER ERRONE ***

PAGE NUMERO :  855    ****

SS-PAGE  28 AV  1888537427      PAGE SUIVANTE  0        SS-PAGE SUI
*** ERREUR *** SS-PAGE VIDE

*** TETE DE CHAINE ***

PAGE NUMERO :  781    ****

SS-PAGE  22 AV  1888537434      PAGE SUIVANTE  1163     SS-PAGE SUI

         0889007D  03C3D6C4  08430070  017E4040         0C81007D

*** SQUATTER ERRONE ***

PAGE NUMERO :  1145   ****

SS-PAGE  0  AV  1881819674      PAGE SUIVANTE  1080     SS-PAGE SUI

         0C040001  047AC5E7  D7404040  10890001         14C1D5C9

***  STATISTIQUES ETABLIES SUR LA BASE : ADMITBD ***

         ESPACE DES  FICHIERS
IL EXISTE DES SQUATTERS NON CHAINES  <-- Există „squatter" neînlă
NB SS-PAGES OCCUPEES : 70974        SUR : 95968       ( 73%)
NB SQUATTERS : 16244        SUR : 70974        ( 22%)
NB SS-PAGES DONT LE HAUT EST VIDE : 97851        SOIT : 318484
NB SS-PAGES DONT LE BAS  EST VIDE : 9660         SOIT : 120993
NB DE CHAINES A  I  ELEMENTS :   44308      SOIT :  0        CH
NB DE CHAINES A  2  ELEMENTS :   6777       SOIT :  6591     CH
NB DE CHAINES A  3  ELEMENTS :   2078       SOIT :  3913     CH
NB DE CHAINES A  4  ELEMENTS :   1021       SOIT :  2897     CH
NB DE CHAINES A  5  ELEMENTS :   414        SOIT :  1582     CH
NB DE CHAINES A  6  ELEMENTS :   84         SOIT :  399      CH
NB DE CHAINES A  7  ELEMENTS :   25         SOIT :  142      CH
NB DE CHAINES A  8  ELEMENTS :   3          SOIT :  17       CH

POUR L'EXPLORATION DES CHAINES IL Y A EU
 44492       FOIS  0  CHANGEMENTS DE PAGE
 6847        FOIS  1  CHANGEMENTS DE PAGE
 1977        FOIS  2  CHANGEMENTS DE PAGE
 938         FOIS  3  CHANGEMENTS DE PAGE
 363         FOIS  4  CHANGEMENTS DE PAGE
 72          FOIS  5  CHANGEMENTS DE PAGE
 19          FOIS  6  CHANGEMENTS DE PAGE

 54123       FOIS  0  CHANGEMENTS DE CYLINDRE
 563         FOIS  1  CHANGEMENTS DE CYLINDRE
 7           FOIS  2  CHANGEMENTS DE CYLINDRE
 14          FOIS  3  CHANGEMENTS DE CYLINDRE
 1           FOIS  4  CHANGEMENTS DE CYLINDRE

VIDAGE PARTITION 02   <-- Prelucrare întreruptă prin intervent

ETAT PROGRAMME                       00023398     01140000

REGISTRES             R0            R1            R2          R
                   F2F0F240      10FFFFFF      000288AC     0028

                      R8            R9            R10         R
                   000233FC      00021720      00022BD0     E834
```

ANEXA 7 **(continuare partea a II-a)**

```
VANTE 28
04C1E5C5  C3404040                    *EI 'ICATEGORIA  <* 'DAVEC    *

VANTE 0

VANTE 0
02E9F15E  00000001                    *HI 'CCODHC 'A=  <A '8Z1;   A    *

VANTE 18
40E50503  08C9D4C5                    *<D AD:EXP   *I AMANI-VECHIME    *
```

ntuite

```
 OCTETS
 OCTETS
ANGEMENTS DE PAGES ET  0         CHANGEMENTS DE CYLINDRE
ANGEMENTS DE PAGES ET  429       CHANGEMENTS DE CYLINDRE
ANGEMENTS DE PAGES ET  60        CHANGEMENTS DE CYLINDRE
ANGEMENTS DE PAGES ET  8C        CHANGEMENTS DE CYLINDRE
ANGEMENTS DE PAGES ET  49        CHANGEMENTS DE CYLINDRE
ANGEMENTS DE PAGES ET  5         CHANGEMENTS DE CYLINDRE
ANGEMENTS DE PAGES ET  C         CHANGEMENTS DE CYLINDRE
ANGEMENTS DE PAGES ET  C         CHANGEMENTS DE CYLINDRE
```

le operator cu: SOCORI ; ABT

```
   3         R4         R5         R6         R7
0000      60580792   622A2518   00028808   622A2518

  11         R12        R13        R14        R15
240C      700A0004   00020E08   00023244   00020164
```

ANEXA 8

MESAJE DE EROARE

Formatul mesajelor : *ER.txxNL n... ⟨*şir de caractere*⟩ ...⟨*mesaj*⟩

xx — codul erorii în hexazecimal ;

n — numărul liniei sursă la care a fost detectată eroarea ;

⟨*şir de caractere*⟩ — şirul de caractere care trebuie să fie realizat ;

t : în funcţie de procesorul apelat are valoarea :

— D pentru compilatorul de structură ;

— R pentru procesorul de cereri (requete).

ERORI LA COMPILAREA STRUCTURII

*ER.D00 NL n...xx...INTERDIT identificator reutilizat în acelaşi bloc ;

*ER.D01 NL n...xx...AU LIEU DE FIN lipseşte un FIN după un bloc sau entitate ;

... ...

*ER.D02 NL n...xx...AU LIEU DE NOMBRE lipsă margine interval de variaţie la definirea unei caracteristici numerice ;

*ER.D03 NL n...xx...AU LIEU DE A lipseşte termenul „A" între numerele care definesc marginile intervalului de variaţie ;

*ER.D04 NL n... xx. ..INTERDIT eroare de sintaxa la definirea unei liste de valori (lipsa „("sau prezenţa valorilor „U" sau „NON" etc.) ;

*ER.D05 NL n...xx...INTERDIT numărul de valori definite în listă este mai mare decît cel declarat ;

*ER.D06 NL n...xx...AU LIEU DE TYPE DE CARAC lipseşte tipul caracteristicii la definire ;

*ER.D07 NL n...xx...INTERDIT numărul entităţilor imbricate este mai mare de 7 ;

*ER.D08 NL n...xx...AU LIEU DE NOM identificator omis sau incorect ;

*ER.D09 NL n...xx...AU LIEU DE DEBUT comanda de definire incorectă (diferită de DEBUT, D sau S) sau lipsă DEBUT pentru bloc, entitate sau formal ;

*ER.D0A NL n...xx...INTERDIT numărul maxim de elemente declarat pentru o listă de valori este negativ sau superior lui 65535 sau nul ;

*ER.D0B NL n...xx...INTERDIT eroare de sintaxă la definirea unei caracteristici valoare numerică ;

*ER.D0D NL n...xx...INTERDIT eroare de sintaxă la definirea unei caracteristici de tip inel sau referire ;

*ER.D0FNNL n...xx...SYSTEME oprirea compilării înaintea detectării „?" (prea multe FIN) ;

*ER.D10 NL n...xx...AU LIEU DE (lipseşte)" după (număr..." ;

*ER.D11 NL n...xx...AU LIEU DE NOMBRE numărul maxim de realizări pentru o entitate este superior lui $(2^{24}-1)$ sau lipseşte ;

*ER.D12 NL n...xx...INTERDIT eroare la definirea unei caracteristici de tip text (lungimea liniei ≠ de 60 sau numărul de linii declarate superior lui 255) ;

*ER.D13 NL n...xx...INTERDIT lungimea declarată pentru un element valoare dintr-o listă de valori >30 ;

*ER.D14 N1 n...xx...INTERDIT citarea referinței sau a caracteristicii de tip invers incorectă ;

*ER.D15 NL n...xx...INTERDIT prea multe imbricări la citarea referinței (restricție sistem) ;

*ER.D16 NL n...xx...INTERDIT citare entitate nerezolvabilă (identificatorul este necunoscut ; xx – identificatorul caracteristicii care referă) ;

*ER.D17 NL n...xx...INTERDIT o caracteristică de tip ⟨inel⟩ este citată de mai multe referințe ;

*ER.D18 NL n...xx...AVEC eroare în sintaxa de definire a unei caracteristici de tip cheie ;

*ER.D1A NL n...xx...FORMAL o caracteristică formal trebuie să fie definită fie la nivel extern structurii, fie în altă caracteristică formal ;

*ER.D1B NL n...xx...FORMAL eroare tip caracteristică formal ;

*ER.D1C NL n...xx...INTERDIT valoarea atribuită unei caracteristici de tip MOT nulă sau superioară lui 80 ;

*ER.D1D NL n...xx...FORMAL valoarea atributului lungime a unei caracteristici de tip FORMAL lipsește, este negativă sau prea mare ;

*ER.D1E NL n...xx...SYSTEME formatul intern construit de sistem depășește 64 K-cuvinte (structură prea mare) ;

*ER.D1F NL n...xx...SYSTEME structura definită depășește capacitatea virtuală a sistemului ($2^{31}-1$) ;

*ER.D20 NL n...xx...DEFINITION...INTERDIT definirea nu este permisă în cazul accesului simultan la baza de date ;

*ER.D24 N1 n...xx...INTERDIT definirea inelului nu se poate face decît în cadru unei entități ;

Detectarea oricăreia dintre aceste erori provoacă, la terminarea compilării, editarea mesajului ERREUR EN DEFINITION iar codul intern rezultat nu este stocat în spațiul real.

ERORI LA COMPILAREA CERERILOR

*ER.R12 NL n...xx...INTERDIT „BLOQUER" și „LIBERER" trebuie să constituie un nivel de bloc în care se intră prin BLOQUER și se iese obligatoriu prin LIBERER, de exemplu :

POUR BLOQUER FIN LIBERER
FAIRE BLOQUER REFAIRE LIBERER FIN
BLOQUER SUIVANT LIBERER
BLOQUER SORTIE LIBERER
BLOQUER SI ALORS LIBERER FIN sînt utilizări interzise ;

*ER.R13 NL n...xx...INTERDIT „EXT", M... = EXT, EXEC ⟨nume program⟩ interzise în blocul BLOQUER LIBERER ;

*ER.R1E NL n...xx...SYSTEME mărimea codului intern construit pentru program depășește 64 K-octeți ;

*ER.R20 NL n...xx...INTERDIT cod de comandă eronat ; EXT interzisă într-un program precompilat ;

*ER.R21 NL n...xx...AU LIEU DE CODE ACTION lipsește codul comenzii pentru o cerere;
*ER.R22 NL n...xx...AU LIEU DE FIN lipsește FIN după o cerere compusă introdusă de POUR, FAIRE sau SI;
*ER.R23 NL n...xx...AU LIEU DE = lipsește „=“ într-o cerere de punere la zi;
*ER.R24 NL n...xx...INTERDIT „REFAIRE“ în afara unui ciclu „FAIRE“; „SORTIE“ fără „POUR“ sau „FAIRE“;
*ER.R25 NL n...xx...AU LIEU DE ALORS lipsa ALORS după SI;
*ER.R26 NL n...xx...INTERDIT eroare în creare sau generare;
*ER.R27 NL n...xx...AU LIEU DE PART DRT partea dreaptă a unei generări sau puneri la zi incorectă sintactic;
*ER.R28 NL n...xx...INTERDIT expresie numerică incompletă;
*ER.R29 NL n...xx...INTERDIT eroare la utilizarea parantezelor în expresii aritmetice;
*ER.R2A NL n...xx...INTERDIT eroare de sintaxă într-o cerere de interogare sau tip de caracteristică neconformă cu cererea;
*ER.R2B NL n...xx...INTERDIT cererea nu se termină prin „?“ (imbricare incoerentă a blocurilor „POUR“, „SI“, „FAIRE“);
*ER.R2C NL n...xx...INTERDIT eroare într-o cerere de interogare cu format (lipsa unui număr sau a unei „)“ în dreapta);
*ER.R2D NL n...xx...INTERDIT identificatorul unei zone de tip „BUFFER“ nu este urmat de număr;
*ER.R2E NL n...xx...INTERDIT eroare la extragerea unui sublanț din Z_i;
*ER.R2F NL n...xx...INTERDIT eroare de sintaxă la LIRE sau ECRIRE;
*ER.R30 NL n...xx...INTERDIT „EXEC“ pe un program nedefinit;
*ER.R31 NL n...xx...AU LIEU DE NOM lipsa unui identificator într-o citație;
*ER.R32 NL n...xx...INTERDIT citația formală a unei entități introduse de un alt termen decît „UN“;
*ER.R33 NL n...xx...AU LIEU DE DEBUT CITATION început citație incorect;
*ER.R34 NL n...xx...INTERDIT termenul TOUT interzis în această citație;
*ER.R35 NL n...xx...AU LIEU DE NOM lipsă identificator după AVEC sau PAR;
*ER.R36 NL n...xx...INTERDIT lipsa unei variabile X_i într-o cerere de definire a unui X_i sau într-o citare de formal;
*ER.R37 NL n...xx...redefinirea incorectă a unui X_i;
*ER.R38 NL n...xx...INTERDIT un astfel de filtru nu este implementat (EN-COURS);
*ER.R39 NL n...xx...AU LIEU DE ; lipsă „;“ la sfîrșitul unui filtru introdus de „AVEC“, „AYANT“, „TELQUE“, „PAR“;
*ER.R3A N1 n...xx...AU LIEU DE .OPERAT COMPAR. lipsă „=“ într-un filtru AVEC; lipsă operator comparație într-o condiție;
*ER.R3B NL n...xx...AU LIEU DE .PART DRT FILT. partea dreaptă a unui filtru este sintactic incorectă;
*ER.3RC NL n...xx...AU LIEU DE .PART DRT FILT. partea dreaptă a unui filtru AVEC incorectă sintactic;
*ER.R3E NL n...xx...INTERDIT „DEPUIS“ trebuie să fie urmat de o valoare numerică;

*ER.R3F NL n ... xx'...INTERDIT eroare sintactică în „SUIVANT" ;

*ER.R40 NL n COMPAR .ENTITE comparările de adrese admise sînt limitate la = sau ┐= ;

*ER.R41 NL n COMPAR .ENTITE în filtre sau puneri la zi dacă partea dreaptă este de tip entitate ; partea stîngă trebuie să fie la fel ;

*ER.R42 NL n COMPAR .ENTITE într-un filtru sau punere la zi, comparație de adresă de realizare etc. entitățile citate în stînga și dreapta trebuie să fie formal identice ;

*ER.R43 NL n COMPAR.NUMERIQUE comparare (sau punere la zi) a unui Y_1 cu o caracteristică alfanumerică ;

*ER.R44 NL n COMPAR. în $\left\{\begin{matrix}\text{LIRE}\\ \text{ECRIRE}\end{matrix}\right\}$ X_1 ; în X_1 mărimea entității din spațiul virtual este foarte mare în comparație cu bufferul în memoria centrală ;

*ER.R45 NL n COMPAR numărul indicat pentru un buffer este prea mare ;

*ER.R46 NL n COMPAR NUMERIQUE într-un filtru sau punere la zi, partea dreaptă fiind numerică, partea stîngă nu este conformă ;

*ER.R47 NL n COMPAR.FORMAL într-o punere la zi, partea stîngă fiind o caracteristică formal, dreapta nu este conformă ;

*ER.R48 NL n COMPAR. NUMERIQUE nu sînt respectate limitele la compararea (punerea la zi) a unei caracteristici numerice cu alta ;

*ER.R49 NL n COMPAR.ALPHA comparare sau punere la zi a unei variabile Z_i sau valoare alfanumerică cu o valoare neconformă sau citația introduce o valoare nonconformă ;

*ER.R4B NL n, COMPAR.BLOC OU TEXTE în filtre sau puneri la zi sînt interzise citările caracteristice bloc sau text ; lipsă nume bloc după EXISTE sau PAS ;

*ER.R4C NL n COMPAR.NUMERIQUE partea stîngă și partea dreaptă a unui filtru sau punere la zi nu sînt de același tip ; NUMDE utilizează cu o caracteristică al cărei tip diferă de : entitate, listă de valori invers sau X_i ;

*ER.R4D NL n COMPAR.VALUER eroare la compararea cu o valoare din listă ;

*ER.R4E NL n COMPAR.FORMAL caracteristică care introduce o citație de formal nu este conformă ; reutilizarea unui buffer nedefinit în program ;

*ER.R4F NL n COMPAR.AVEC partea stîngă și partea dreaptă a filtrului nu sînt conforme ;

*ER.R50 NL n CONTEXTE xx NON DEFINI DANS yy caracteristica ⟨xx⟩ nu este definită structural în contextul ⟨yy⟩ ;

*ER.R51 NL n CONTEXTE X_i, REDEFINI redefinirea lui X_i citat este interzisă deoarece contrazice o definire precedentă aflată la un nivel superior ;

*ER.R52 NL n CONTEXTE X_i NON DEFINI X_i citat nu este definit ;

*ER.R53 NL n CONTEXTE STRUCTURE interogațiile simple se aplică la orice tip de caracteristică în afara celor de tip bloc și formal ;

*ER.R54 NL n CONTEXTE STRUCTURE caracteristica utilizată ca „tată" nu este definită astfel în structură ;

*ER.R55 NL n CONTEXTE AVEC caracteristicile aflate după AVEC sau PAR nu sînt declarate cu acces direct ;

*ER.R56 NL n CONTEXTE INVERSE „G" cu partea dreaptă nu este admis decît pentru INVERSE ; citația din partea dreaptă la generarea unui invers nu este conformă celei asociate caracteristicii inverse la definire ;

*ER.R57 NL n CONTEXTE SE „SE" (suprimarea ansamblului) nu este permisă decît pentru inel sau invers ; „S" (suprimare) se aplică numai la entitate, invers sau referință ; „C" (creație) nu se aplică decît la entități ;
*ER.R58 NL n CONTEXTE xx NON DEFINI DANS yy structura nu există ;
*ER.R59 NL n CONTEXTE SYSTEME eroare sistem ; codificare internă transmis eronat ;
*ER.R5A NL n CONTEXTE SYSTEME eroare sistem : codificare internă eronată;
*ER.R5B NL n CONTEXTE PROGRAMME după execuția unui „EXEC" X_i reale și formale nu sînt confirmate ; la sfîrșitul programului X_i locale nu sînt puse la nedefinit ; contextul de execuție al programului nu este conform ;
*ER.R5C NL n CONTEXTE PROGRAMME eroare de context la execuția unui program.

ERORI LA DEFINIRE MACRO/PRECOMPILATE/IMT

Formatul mesajelor :

ERR.MACRO xxx NL n .:. ⟨șir caractere⟩
xxx : codul zecimal al erorii ;
n : numărul liniei în textul sursă ;
⟨*șir caractere*⟩ : șirul de caractere analizat.

1) *Definire macro* :

101 numele macro nu trebuie să fie un cuvînt rezervat sau numele altui program ;
102 numele atribuit macro a fost atribuit unui program precompilat ;
103 numele atribuit macro a fost atribuit unui program IMT ;
104 macro deja definită (se poate redefini numai după execuția unui : SUPEXP) ;
105 : FDEF interzis în modelul de apel ;
106 ? interzis în modelul de apel ;
107 numărul de parametri definiți în modelul de apel⟩ decît numărul maxim parametri (16) ;
108 absență nume program IMT după PW ;
109 absență EXP după numele programului de control ;
110 cuvînt cheie sau ? interzis în expansiune ;
111 numărul de parametri definiți în expansiune este mai mare decît cel declarat în modelul de apel ;
112 un parametru nu poate fi utilizat de două ori în modelul de apel ;
113 expansiunea macro depășește limita de 128 Ko ;
132 macro redefinită la o altă consolă în timpul DEFMAC pentru aceasta ;

2) *Apel macro*

201 separator de apel non conform modelului de apel ;
202 ? într-un parametru de apel al unei macro ;

3) *Definirea programelor precompilate*

501 un cuvînt cheie sau rezervat nu poate fi atribuit ca nume pentru program ;
502 numele unui program nu poate fi acela atribuit unui macro ;
503 un nume de program IMT nu poate fi atribuit unui program precompilat ;
504 program precompilat definit (nu a fost făcut inactiv prin SUPEXP) ;
505 eroare definire program ; EXP absent sau plasat greșit ;

508 ? interzis la definirea contextului programului ;
509 numele care urmează lui PW nu este al unui program IMT ;
510 cuvînt cheie sau ? interzis în expansiune ;
511 lipsă PW, CONTXT sau EXP după numele programului ;
521 eroare la compilarea programului ;
522 program definit de o altă consolă activă în timpul execuţiei DEFPRO pentru acesta ;
531 saturarea numărului de apeluri diferite ale macro, programe precompilate şi IMT ;

4) *Definirea programelor IMT*

801 un cuvînt cheie sau rezervat nu poate fi atribuit ca nume pentru un program IMT ;
802 un nume de macro nu poate fi atribuit unui program IMT ;
803 un nume de program precompilat nu poate fi atribuit unui program IMT ;
804 program IMT definit (nu a fost făcut inactiv prin SUPEXP) ;
805 un nume de program IMT nu poate avea mai mult de 8 caractere ;
806 crearea programelor IMT interzisă în conversaţional ;
811 numele programului de ordinul IMT trebuie să fie cel declarat în LINK ;
812 utilizarea SGF interzisă ;
813 program IMT prea mare (max. 64 ko) ;
814 un program IMT nu poate fi segmentat ;
815 eroare pe numărul de secvenţă al liniilor IMT ;
816 eroare pe cheia de control (linie IMT invalidă)
817 linia PRCGIN absentă sau decalată
818 linia TABSEG absentă sau decalată ;
819 linia SEGNAME absentă sau decalată ;
820 prezenţa unei linii eronate la începutul sau în cadrul segmentului (cod bloc segment absent) ;
821 un program IMT trebuie să fie relocatabil ;
822 interzisă utilizarea unui program IMT vid ;
851 linia FIN IMT absentă ;

5) *Ediţie şi suprimare*

301 numele care urmează lui SUPEXP nu este al unui macro, PRO sau IMT ;
302 numele care urmează lui SUPMAC nu este al unui macro ;
303 numele care urmează lui SUPPRO nu este al unui program precompilat ;
304 numele care urmează lui SUPIMT nu este al unui program IMT ;
311 numele care urmează lui EDIMAC nu este al unui macro ;
312 numele care urmează lui .EDIPRO nu este al unui program precompilat ;

6) *Saturarea spaţiilor*

950 saturarea spaţiu macro (s-a atins numărul maxim de macro şi programe precompilate) ;
951 saturarea spaţiu program (s-a atins numărul maxim de programe precompilate şi IMT) ;
952 saturarea dicţionar pentru nume macro şi programe ;
955 eroare sistem.

7) *Erori la execuție*

COD	MESAJ EROARE
01	FIN FICHIER
03	REALISATION DEJA PRESENTE
06	ENTITE SATUREE
07	RANG ABSOLU EN DEHORS ENTITE
09	REALISATION ENTITE ABSENTE
0E	CLE DEJA PRESENTE
0F	CLE NON TROUVEE
52	VALEUR NUMERIQUE ERRONEE
54	VALEUR DE LISTE ERRONEE
80	LANCEMENT PROGRAMME IMPOSSIBLE
90	PROGRAMME PERIME
FF	ARGUMENT DS TROP PETIT

8) *Erori sistem*

Sintaxă mesaj

***** ERREUR NNUMERO ⟨n⟩

⟨n⟩ numărul erorii

– **erori generale :**

* IMPOSSM * partiție prea mică

1 eroare depi (valoare prea mare) (DEPI)

2 eroare empi (valoare prea mare) (EMPI)

3 numărul de cadre în memoria secundară prea mic

5 cod operație necunoscut (la COM)

6 eroare salt interpretat (BRIN)

8 saturare tabela programelor precompilate

12 lipsă comanda $ (inițializare TRACE)

16 eroare în VREP

20 absența comenzi .OPTION (FORMAL sau DS prea mic)

– *erori la utilizarea SGT* :

30 LCB activ pe altă sarcină

31 numărul de sarcini active prea mare

32 eroare necunoscută pe ACTP

– *erori la paginare* :

100 eroare parametru tip

101 eroare creare sub-pagină (incoerență fizică) ;

102 eroare calcul sub-pagină reală

103 eroare lungime lanț discriminant/rapid

– *erori la utilizarea benzilor magnetice* :

200 sens intrare-ieșire incompatibil cu PRM

201 lipsă TDF pentru fișier

202 eroare înlănțuire TDF pe sarcină

203 eroare la definirea LCB

204 incompatibilitatea între lungimea cerută și RCS în cazul RCF=FIX

9) *Erori la utilizarea interfeței*

— erori la deschiderea (CALL SOPEN) : mesajele sînt precedate de ** ERR SOPEN.

TROP D'OUVERTURE : mai mult de 20 SOPEN active simultan ;

FICHIER DEJA OUVERT : doua SOPEN pentru un utilizator fără SCLOSE ;

TROP DE Xi DEMANDE : numărul de variabile Xi negativ (NRx)

PAS ASSEZ MEMOIRE : nu a fost rezervat spațiu suficient pentru utilizatorii simultani (UN/OPTION)

PAS ASSEZ MEMOIRE (OPTION/DS) : nu a fost rezervată suficientă memorie dinamică

BASE INCONNUE : baza inexistentă în baza de baze

UTILIZATEUR INCONNUE : utilizator necunoscut

DISQUE ABSENT : eroare SGF la deschiderea fizică a bazei

— erori la execuție (RUN) : mesajele sînt precedate de ** ERR SGBD.

FICHIER NON OUVERTE : nu s-a făcut SOPEN pe baza respectivă

LONGUEURE/TROP GRANDE : lungimea bufferului nu poate depăși 32 k octeți

ADRESSE BUFFER TROP GRANDE : adresa bufferului este prea mare (superioară lui 512 kocteți)

LONGUEURE BUFFER NON MULTIPLE DE MOT : lungimea unui buffer de comunicare trebuie să fie multiplu de cuvînt

NOMBRE DE PROGRAMME INCOHERENT : numărul programelor (NRPGM) este negativ

NOMBRE DE Xi INCOHERENT : NPC sau NX negativ

TROP DE Xi : NPC incoerent (>9)

NOMBRE DE Xi INCOHERENT : numărul variabilelor Yi (NY) negativ

TROP DE Xi : NY superior numărului admisibil în zona unui utilizator

Yi TROP GRAND : Yi trebuie să nu depășească valoarea maximă reprezentabilă într-un cuvînt binar

Yi NON NUMERIQUE : Yi nu sînt definite cu clauza COMP-3

NOMBRE DE Zi INCOHERENT : numărul variabilelor Zi negativ

TROP DE Zi : numărul variabilelor Zi este superior maximului admis de SOCRATE

PROG. NON RECONNU : un program indicat la CALL SGBD nu există

AU LANCEMENT D'UN PROGRAMME : eroare de context, program definit eronat

FATALE AU RUN : lungimea buff. de comunicație este inferioară lungimii formalului pe care dorim să o placăm.

10) *Erori ale limbajului de comandă*

*** ERREUR *** MOTICLE ⟨nume cuvînt cheie⟩ — utilizarea unui cuvînt cheie eronat —

*** ERREUR *** ARGUMENT ⟨nume argument eronat⟩ — utilizarea unui argument eronat —

*** ERREUR *** SEQUENCE CARTE % — secvența linie de comandă eronată

*** ERREUR CARTE % LUE L'UTILIZATEUR — citirea unei comenzi % de către utilizator —

Aceste erori provoacă oprirea prelucrării și închide fișierele (bazele de date).

Teste

(continuare de la paginile 15–16)

10. Se scriu următoarele instrucţiuni. Undeva s-a produs o greşeală. Unde este această greşeală şi ce se întîmplă la executarea acestor instrucţiuni :

```
I(30) 'ABSOLVENTII FACULTATII (SC'
I(57) 'OLII: ' I (44) Y1 ECRIRE
```

Răspuns : Greşeala este pe linia a 2-a, pentru că se încearcă scrierea codului (Y1) din coloana 44, în loc de 63, de exemplu (adică se face suprapunerea de scriere).

Pe listing se va scrie astfel :

```
ABSOLVENTII F 222222II (SCOLII:
```

11. Fie următorul program de încărcare :

```
1  D X1    D   X2   D  X3   D   X4
2  M Y2 = 0  M  Y3 = 0  M  Y4 = 1  M  Y5 = U
3  M Y1  = 100
4  M X1  = FORMAL FCOPIL DANS BUFFER 0
5  FAIRE
6        LIRE DANS BUF 0
7        M Y3 = TC DE X1
8        SI  Y3 = Y1 ALORS I (5) ' TC NU ESTE 100'
9                          ECRIRE
10                         REFAIRE
11                  SINON
12       M Y2 = MARCA DE X1
13       SI Y2 = 99999 ALORS SORTIE FIN
14       M Y5 = ERCOP DE X1
15       SI EXISTE UN LUCRATOR X2 AVEC MARCA = Y2
16                          ALORS
17       G UN COPIL Y5 DE X2
18       FAIRE
19       SI Y4 > Y5 ALORS SORTIE FIN
20       M  X4 = FORMAL FCO Y5 DE X1
21       M  X5 = UN COPIL Y5 DE X2
22       M  NUME-C DE X5 = FNUME DE X4
23       M  ANI DE X5 = FANI DE X4
24       REFAIRE FIN
25       REFAIRE FIN
26       ?
```

Întrebare : Ce erori de sintaxă s-au produs ?

Răspuns : Pe linia 15 trebuie după AVEC MARCA = Y2 să se pună ; (este vorba de un filtru). Mai trebuiesc şi doi de FIN pentru a „închide" cei doi de „SI".

12. Se scriu următoarele instrucţiuni :

```
             PENTRU TOTI LUCRATOR X7
                ATRIBUIE Z1 = SEX DE X7
                   ATRIBUIE Y1 = ANUL-N DE X7
             DACA Z1 = 'MA' SI EXISTA Y1 ATUNCI
DACA Y1 = 1928 ATUNCI M Y2 = Y2 + 1 DACA-NU
DACA Y1 = 1938 ATUNCI M Y3 = Y3 + 1 DACA-NU
DACA Y1 = 1943 ATUNCI M Y4 = Y4 + 1 DACA-NU
DACA Y1 = 1948 ATUNCI M Y5 = Y5 + 1 CACA-NU
DACA Y1 = 1953 ATUNCI M Y6 = Y6 + 1 DACA-NU
DACA Y1 = 1957 ATUNCI M Y7 = Y7 + 1 DACA-NU
DACA Y1 = 1960 ATUNCI M Y8 = Y8 + 1 DACA-NU
FIN
     FIN FIN FIN FIN FIN  FIN    FIN
```

Întrebare : Ce greşeli s-au făcut.

Acest grup de instrucţiuni vor funcţiona ?

Dacă nu : *de ce ?*

Dacă da : *în ce situaţie.*

Răspuns : — Nu s-a produs nici o greşeală.

— Acest grup de instrucţiuni vor funcţiona în cazul în care cuvintele rezervate SOCRATE (POUR,M şi ALORS etc.) vor fi „traduse" în cuvinte româneşti, cu ajutorul MACROGENERATORULUI.

13. Pentru a obţine o anumită situaţie se scriu următoarele instrucţiuni :

```
1   POUR TOUT CADRE X2
2     DE UN SECTIE      X3
3         AVEC COD = Y1
4               PAR MARCA
```

Întrebare : *Sînt erori de sintaxă ?*
Răspuns : Da. Liniile 3 şi 4 trebuiau scrise astfel :

```
AVEC COD = Y1 ;
PAR MARCA ;
```

14. Într-un program de încărcare instrucţiunile apar astfel :

```
 1   M    X1 = FORMAL FLUCRARE DANS BUF 0
 .
 .
 .
 7   M      Y2 = MARCA DE X1
 8   SI     Y2 = 99999 ALORS SORITE
 9   SI EXISTE UN LUCRATOR X2 AVEC MARCA = Y2 ;
10   ALORS
11          G UN LUCRĂTOR X3
12          M TIP DE X3 = TIP DE X1
13          M DENUMIRE DE X3 = DENUMIRE DE X1
14          M AM DE X3 = AN DE X3
```

Întrebare : *S-a strecurat vreo eroare de sintaxă ?*
Răspuns : Da. Pe linia 8 trebuia scris

```
Si Y2 = 99999 ALORS SORTIE FIN
```

iar pe linia 14 :

```
M AN DE X3 = AN DE X1.
```

15. Într-un program de încărcare se scrie :

```
 1  M   Y2 = 0 M Y3 = 0 M Y4 = 0 M Y5 = 0
 .      .
 .      .
 .      .
 7      G UN CURS X3
 .      .
 .      .
 .      .
15  M   Y4 = Y4+1
 .      .
 .      .
 .
20      I(+2) 'AU FOST GENERATE'
21      I(+2) Y4
22      I(+2) 'REALIZARI DE ENTITATE CURS'
23      ECRIRE
24   ?
```

Întrebare : *S-a produs vreo eroare de sintaxă ?*
Răspuns : Da. Pe linia 15 dă ER.R50 :

```
Y4+1 NON DEFINI DANS FICHIER
```

Observaţie : Se întîmplă acest lucru cînd Y4+1 se scrie fără spaţiu în stînga şi dreapta lui „+". Dacă se scrie

```
M  Y4 = Y4 + 1
```

este corect.

16. Această instrucţiune este corect scrisă ?

```
1    SI EXISTE UN LUCRATOR X1 AYANT
2         GR-SANG = '0' ;
3     ALORS
4         I NUME
5     FIN
6  ?
```

Răspuns : Nu. Linia 4 trebuie scrisă astfel :

```
I NUME DE X1
```

17. Se scrie instrucţiunea

```
1      SI EXISTE UN LUCRATOR AYANT
2            GR-SANG = 'G' ;
3      ALORS
4            I UN CALIFICATIV DE X1
5      FIN
6      ?
```

Întrebare : *Ce greşeli s-au strecurat.*

Răspuns : Linia 1 trebuie scrisă astfel :

```
SI EXISTE UN LUCRATOR X1 AYANT ...
```

Observaţie : În astfel de cazuri se folosesc calificatori de tip variabilă X1.

18. Se scrie următoarea cerere SI :

```
1    SI EXISTE UN ANGAJAT X1 AYANT MARCA > 5000 ;
2    ALORS
3    I(1) MARCA DE X1
4    I(10) NUME DE X1
5    SINON
6    I(1) 'NU EXISTA MARCA MAI MARE CA 5000'
7    FIN
8    ?
```

Întrebare : Ce erori s-au strecurat ?

Răspuns : După comenzile I cu format trebuie dată comanda ECRIRE, adică după liniile 4 şi 6.

19. Se dă următoarea structură :

```
D ENTITE 5000 LUCRATOR
     DEBUT
          .
          .
          .
       caracteristici —
          .
          .
          .
  ENTITE   10 COPIL
     DEBUT
       caracteristici
          .
          .
          .
          .
     NEAM REFERE RUDA DE UN LOC-MUNCA
  FIN
  ENTITE    50 LOC-MUNCA
     DEBUT
          .
          .
          caracteristici
          .
          .
     RUDA ANNEAU
     FIN
   FIN
  FIN
     ?
```

Întrebare : Ţinînd seama că pentru fiecare entitate s-a definit cîte un FORMAL, în vederea încărcării bazei de date, să se specifice care va fi ordinea de rulare a programelor de încărcare.

Răspuns : După modul cum se face legătura între entităţi (REFERE şi ANNEAU), rezultă că ordinea de rulare a programelor de încărcare va fi :

— programul de încărcare a entităţii LOC-MUNCA
— programul de încărcare a entităţii LUCRATOR
— programul de încărcare a entităţii COPIL

Observaţie : Entităţile nu se încarcă în ordinea declarată în structură, ci după modul de „referire" a uneia faţă de celelalte.

20. Pentru a obţine un anumit raport din baza de date se scrie următorul program :

```
POUR TOUT LUCRATOR X1 PAR NUME AYANT
        JUD-N ; = ARGES' ; ;
            I (10) LOC-N DE X1
            I (34) ANUL-N DE X1
        ECRIRE
    FIN
    ?
```

Întrebare : S-a folosit corect filtrul cu AYANT şi criteriul de triere cu PAR ?
Răspuns : Nu. Trebuia scris :

```
POUR TOUT LUCRATOR X1 PAR NUME ;
                AYANT JUD-N = 'ARGES' ;
```

21. Se doreşte obţinerea unor informaţii din baza de date cu ajutorul instrucţiunii :

```
I PRENUME DE TOUT LUCRATOR AVEC
        NUME = 'TOMESCU' ;
```

În acest caz se foloseşte un acces direct prin criteriu introdus prin cuvîntul cheie AVEC.
Întrebare : În cazul de faţă, cînd este permisă folosirea accesului direct prin criteriu ?
Răspuns : În structură trebuia precizată caracteristica (rapidă sau discriminantă) NUME, cu cheie (AVEC CLE sau AVEC CLE UNIQUE).

22. Se scrie următoarea secvenţă de program pentru obţinerea unor informaţii din baza de date :

```
1  POUR TOUT LUCRATOR X1 PAR NUME ;
2    AYANT JUD-N = 'ARGES'
3    I (10) LOC-N DE X1 ;
4  FIN
5  ?
```

Întrebare : Este corect utilizat filtrul cu AYANT ?
Răspuns : Nu. a) Liniile 2 şi 3 trebuiau scrise astfel :

```
AYANT JUD-N = 'ARGES' ;
I (10) LOC-N DE X1
```

b) lipseşte instrucţiunea ECRIRE.

23. Se dau următoarele variante de comenzi pentru obţinerea unei anumite situaţii :

a)
```
POUR TOUT LUCRATOR X1
        AYANT SEX = 'MA' ; ET JUD-N = 'IALOMITA' ;
I (10) NUME DE X1
I (35) PRENUME DE X1
ECRIRE FIN ?
```

b)
```
POUR TOUT LUCRATOR X1
    AYANT SEX = 'MA' ; ET JUD-N ='IALOMITA'
    I (10) NUME DE X1
    I (35) PRENUME DE X1
    ECRIRE
    FIN
    ?
```

c)
```
POUR TOUT LUCRATOR X1
    AYANT SEX = 'MA' ET JUD-N = 'IALOMITA' ;
    I (10) NUME DE X1
    I (35) PRENUME DE X1
    ECRIRE
    FIN
    ?
```

Întrebare : Care dintre aceste variante este cea bună.
Răspuns : c).
Secvenţa corectă este c) deoarece are filtrul corect construit.

24. Realizaţi un program IMT pentru refacerea contorului şirului de biţi de prezenţă distrus în urma unui incident (hard, de obicei).

Valoarea contorului este detectată utilizînd funcţia D; M Yi = D TOUT <nume entitate> |<calificare>|. Valoarea reală pe care ar trebui să o aibă este obţinută efectuînd teste de existenţă pe şirul biţilor de prezenţă. Modul de detectare al acestei anomalii este prezentat în capitolul 8.

Rezolvare:

Prima realizare a unei entităţi este precedată de două şiruri de biţi cu structura:

|contor|şir biţi suprimare|contor|şir biţi de prezenţă|

Numărul de biţi rezervaţi pentru contor se calculează plecînd de la numărul maxim de realizări declarat pentru entitate (numărul minim de biţi necesari pentru a reprezenta în binar această valoare).

Din programe scrise în LMD-SOCRATE se poate avea acces la adresa primei realizări din entitate cu cereri de forma:

M Xi = UN <nume entitate> 1 |<calificare>|

Pentru poziţionarea pe <contor> este necesar să se scadă din această adresă valoarea dată de numărul de cuvinte rezervat pentru a memora şirul de biţi de prezenţă. Pentru program considerăm că X1 conţine adresa primei realizări din entitate , iar Y1 lungimea în cuvinte a şirului de biţi. În Y2 vom furniza programului dimensiunea în biţi a controlului iar în Y3 valoarea pe care ar trebui să o aibă acesta.

Programul construit este inclus în spaţiul "program" al bazei de date, cu ajutorul macrogeneratorului dintr-o bibliotecă în format executabil (vezi PROGRAMUL 1).

Interpretarea liniilor programului este următoarea:

- liniile 1-6: permit accesul la zonele din BCB (ZONCOM) prin care se transmit argumentele la program;
- liniile 15-19: stabilirea adresei zonei de argumente necesare primitivelor de citire /scriere utilizînd paginarea;
- linia 20: adresa primei realizări de entitate furnizată în X1 este adusă în registrul general 0;
- linia 21: dimensiunea în cuvinte a şirului de biţi este adusă din Y1 în registrul general 1;
- linia 22: prin scădere se află adresa la care găseşte valoare contorului;
- linia 24: apel primitivă de citire a unui cuvînt de la adresa dată de registrul general 0. Cuvîntul citit va fi plasat în registrul general 0. Din acest cuvînt trebuie să salvăm biţii, eventuali, care nu aparţin contorului. Acest lucru se realizează în liniile 25-29;
- liniile 30-31: valoarea contorului plasată în parametrul Y3 este concatenată cu eventualii biţi salvaţi;
- liniile 32-34: cuvîntul obţinut prin aceste operaţii este rescris în baza de date;
- linia 35: reîntoarcere la programul apelant;
- liniile 36-50: descriere primitive de citire scriere şi a tabelei de argumente necesare acestora.

Deşi programul construit are o anumită destinaţie facem următoarele observaţii asupra modului său de funcţionare:

- dacă argumentul Y1 are o valoare negativă atunci poziţionarea sa se va efectua "înainte" faţă de adresa specificată în X1;
- dacă dimensiune contor este 32 atunci practic programul permite scrierea valorii binare furnizate în variabila Y3.

Dacă Y3 este nedefinit (are valoarea 0 binar) programul ne permite să scriem o valoarea nedefinită la adresa specificată (deci putem să ştergem un inel, o referire sau orice altă zonă care conţine inconsistenţe). Programul poate fi utilizat în cazul remedierii anomaliilor detectate la testarea coerenţei bazei de date (capitolul 8).

În general utilizarea acestui gen de program este rezervată administratorului bazei de date. Această rezervare este dictată şi de faptul că o proastă utilizare poate duce la consecinţe nefavorabile pentru baza de date. De regulă acest gen de program este considerat ca "primitivă", programelor apelante revenindu-le sarcina să testeze parametrii de apel, eventual într-un context bine precizat, şi să semnaleze inconsistenţele acestora.

Vom prezenta în PROGRAMUL 2 şi PROGRAMUL 3 utilizarea programului REFCONT pentru refacerea contorului şirului de biţi de prezenţă al unei entităţi (VALI) imbricate într-o altă entitate (VERT). Structura logică a acestor entităţi este:

*** Programul 1 este la pag. 490–492.**

```
ENTITE 16 VERT
 DEBUT
 .
 .
 .
ENTITE 10000 VALI
 DEBUT
 .
 .
 .
 FIN
FIN
```

Şirul de biţi de prezenţă al entităţii VALI ocupă 313 cuvinte iar numărul de biţi pentru contor este 14. În PROGRAMUL 2 este admisă o refacere punctuală a valorii contorului prin specificarea entităţii de nivel superior (VERT). În PROGRAMUL 3 refacerea se efectuează parcurgînd toate realizările entităţii VERT.

Pentru fiecare refacere de contor se deschide (JNLC0) şi se închide o tranzacţie (JNLC1) pentru a asigura coerenţa bazei de date (sînt utilizate primitivele de tranzacţionare SOGER). În cazul întreruperii accidentale sau voite a derulării acestei prelucrări se va efectua reluarea modificînd linia 2 cu DEPUIS <nr>, unde <nr> este valoarea afişată pentru ultima realizare de entitate tratată ***VERT:<nr>.

PROGRAMUL 2

```
*        RUN FN:R

** SOCRATE STARTED ** V1.5 LE:09/02/89 A 07.01.09.08.

 1 D X2
 2 FAIRE
 3     M Y9=EXT
 4     SY Y9=999999 ALORS SORTIE FIN
 5     M X2=UN VERT Y9 /* ADRESA VIRTUALA REALIZARE ENTI */
 6     M Y2=D TOUT VALI DE X2
 7     I (1) 'VALOARE CONTOR:' I (+10 -10) Y2 ECRIRE
 8     M Y1=313 /* DIMENSIUNE IN CUVINTE SIR DE BITI */
 9     M Y2=14 /* DIMENSIUNE CONTOR */
10     M Y3=0 M Y4=1
11     FAIRE
12         SI Y4>10000 ALORS SORTIE FIN
13         SI EXISTE UN VALI Y4 DE X2
14           ALORS
15              M Y3=Y3 + 1
16         FIN
17         M Y4=Y4 + 1 REFAIRE
18     FIN
19     I (1) 'NUMAR REALIZARI DEFINITE:' I (+10 -10) Y3
20     ECRIRE
21     M X1=UN VALI DE X2 /* ADRESA REALIZARII DIN ENTI */
22     EXEC REFCONT
23     M Y1=D TOUT VALI DE X2
24     I (1) VALOARE NOUA CONTOR:' I (+10 -10) Y1 ECRIRE
25     REFAIRE
26 FIN ?
27 44
VAOARE CONTOR.            7645
NUMAR REALIZARI DEFINITE: 7729
VALOARE NOUA CONTOR:      7729
28 999999
```

PROGRAMUL 3

```
%       RUN FN:R

** SOCRATE STARTED ** V1.5 LE:09/02/89 A 07.05.29.78.

D X1 D X2
DEPUIS 0
POUR TOUT VERT X2
 M Y1=NUMDE  X2
 M Y2=D TOUT VALI DE X2
 M Y3=1 M Y4=0
 FAIRE
  SI Y3>10000 ALORS SORTIE FIN
  SI EXISTE UN VALI Y3 ALORS M Y4=Y4 + 1 FIN
  M Y3=Y3 + 1
 REFAIRE
 FIN
 I (1) '***VERT:' I (+3 -2) Y1
 I (+1) 'CONTOR:'
 I (+6 -5) Y2
 I (+1) 'NBIT:' I (+6 -5) Y4
 SI Y4=Y2
 ALORS
      I (+1) '**COERENTA**'
    SINON
      I (+1) '****INCOERENTA****'
      EXEC JNLC0
      M X1=UN VALI 1 DE X2
      M Y1=313 /* DIMENSIUNE SIR BITI PREZENTA VALI */
      M Y2=14 /* LUNGIME CONTOR VALI (BITI)        */
      M Y3=Y4 /* VALOARE REALA PENTRU CONTOR       */
      EXEC REFCONT /* APEL PROGRAM DE REFACERE VALOARE */
      /* REALA CONTOR SIR BITI PREZENTA VALI       */
      EXEC JNLC1
      M Y1=D TOUT VALI DE X2
      I (+1) 'REFACUT LA:' I (+6 -5) Y1
 FIN
 ECRIRE FIN ?
***VERT: 1 CONTOR:   193 NBIT:   193 **COERENTA**
***VERT:  2 CONTOR:  4654 NBIT:  4654 **COERENTA**
***VERT:  3 CONTOR:  1316 NBIT:  1316 **COERENTA**
***VERT:  4 CONTOR:  1118 NBIT:  1118 **COERENTA**
***VERT:  5 CONTOR:   342 NBIT:   342 **COERENTA**
***VERT:  6 CONTOR:  4495 NBIT:  4495 **COERENTA**
***VERT:  7 CONTOR:  9704 NBIT:  9704 **COERENTA**
***VERT:  8 CONTOR:  5014 NBIT:  5014 **COERENTA**
***VERT:  9 CONTOR: 15133 NBIT:  5709 ****INCOERENTA*** REFACUT LA: 5709
***VERT: 10 CONTOR:  2448 NBIT:  2448 **COERENTA**
***VERT: 11 CONTOR:   193 NBIT:   193 **COERENTA**
***VERT: 12 CONTOR:   193 NBIT:   193 **COERENTA**
***VERT: 13 CONTOR:   193 NBIT:   193 **COERENTA**
***VERT: 14 CONTOR:   193 NBIT:   193 **COERENTA**
***VERT: 15 CONTOR:  4528 NBIT:  6229 ****INCOERENTA*** REFACUT LA 6229
***VERT: 16 CONTOR:  5544 NBIT:  5544 **COERENTA**
```

Vom arăta în continuare modalitatea de utilizare a programului REFCONT pentru a şterge conţinutul caracteristicii <fun-spe> din entitatea <funcţii>, realizarea 10. Ştergerea este necesară deoarece am constatat, prin testele de coerenţă logică efectuate, că nu mai există o altă cale de reparare a acestei incoerenţe. Structura secvenţei de ştergere poate fi următoarea:

```
%       RUN FN:R
M X1 = UN FUNCTII 10 /* adresa realizare din entitate
```

```
M Y1 = 0              /* deplasarea primului cuvint in inel          */
M Y2 = 32             /* lungimea tn biti a valorii pe care o scriem */
M Y3 = U              /* valoarea pe care dorim să o scriem          */
  EXEC REFCONT
M Y1 = -1             /* pozitionare "inainte" pe cel de-al 2-lea cuvint */
                      /* al caracteristicii de tip inel              */
  EXEC REFCONT ?
```

Analizînd programul construit nu obţinem o imagine a unui "optim" deoarece fiecare apel tratează un singur cuvînt de 32 de biţi. Considerînd faptul că această operaţie se execută numai în cazuri extrem de rare această utilizare poate fi considerată destul de bună. În cazul în care utilizatorul doreşte să şteargă, de exemplu, mai multe cuvinte consecutive putem să construim macroinstrucţiuni parametrizate. De exemplu, construirea unei macroinstrucţiuni STERG care să primească drept parametrii adresa de unde doresc să încep şi numărul de cuvinte pe care le şterg poate fi făcută astfel:

```
:DEFMAC STERG : :
:EXP
  FAIRE
    SI X1 = U ALORS SORTIE FIN
    M X1 = :1:
    M Y1 = 0
    M Y2 = 32
    M Y3 = U
    M Y4 = :2:
    FAIRE
      SI Y4 = 0 ALORS SORTIE FIN
      EXEC REFCONT
      M Y4 = Y4 - 1
      M Y1 = Y1 - 1
      REFAIRE
    FIN
  FIN
:FDEF ?
```

De exemplu dacă dorim să ştergem realizarea 10 a entităţii <persoana> care are 67 de cuvinte vom efectua următorul apel:

```
%         RUN FN:R
M X2 = UN PERSOANA 10
STERG X2 67 ?
```

25. Să se rezolve parcurgerea unui inel, declarat ca lanţ dublu, în ordine inversă.

Rezolvare:

Un inel declarat cu lanţ dublu ocupă două cuvinte cu semnificaţia PRIMUL|ULTIMUL intrat în lanţ.

Această parcurgere se va efectua exploatînd cîmpul ULTIMUL pentru a afla adresa realizării de entitate care conţine referirea cu lanţ dublu asociată. Referirea cu lanţ dublu ocupă trei cuvinte de memorie cu semnificaţia TATA | URMATOR | PRECEDENT. Pentru a realiza parcurgerea din referire se va exploata cîmpul PRECEDENT pentru a afla adresa următoarei entităţi de prelucrare. Exploatarea se va termina în momentul în care acest cîmp are valoarea nedefinită.

Pentru realizarea acestui scop vom construi două subprograme IMT:

- FROMLAST: acest program primeşte ca parametrii, la intrare, în X1 adresa realizării de entitate care conţine declaraţia de referire iar în Y1 deplasarea acesteia, în cuvinte, în cadrul entităţii. Programul furnizează la ieşire, în X2, valoarea conţinută de cîmpul ULTIMUL al referirii. Dacă această variabilă este nedefinită atunci înseamnă că nu există realizări grupate pe acest inel. Programul realizează de fapt doar o fixare pe prima realizare de entitate din lanţ (vezi PROGRAMUL 4);

- TOFIRST: primeşte la intrare în variabila X2 adresa entităţii care conţine declaraţia de referire iar în Y2 deplasarea acestuia în entitate. Furnizează la ieşire în X2 adresa următoarei realizări grupate pe inel. Dacă X2 are valoarea nedefinit înseamnă

*** Programele 4 şi 5 sînt la pag. 492–494.**

că parcurgerea s-a terminat (vezi PROGRAMUL 5).

Prin cooperarea acestor programe se va realiza, de fapt, o parcurgere FIFO a lanţului.

Exemplu de utilizare a programelor FROMLAST şi TOFIRST

Pentru o persoană dorim să aflăm ce recompense a avut în ordinea acordării acestora. Vom exploata inelul <pers-rec> din entitatea <persoana> şi referirea asociată <recper-rec> din entitatea <recompense>. Pentru fiecare persoană vom imprima numele şi prenumele acesteia şi pentru fiecare recompensă a sa vom imprima denumirea şi data acordării acesteia.

```
M X1 = UN PERSOANA AYANT PERS-REC |= U:
M Y1 = 60 /* deplasarea declaratiei de inel in entitatea persoana */
M X2 = U
I (1) '*' I (+1) NUME DE X1 I (+1) PRENUME DE X1 ECRIRE
FAIRE
  EXEC FROMLAST /* fixare pe prima recompensa */
  SI X2 = U ALORS SORTIE FIN
   M Y2 = 3
   FAIRE
     SI X2 = U ALORS SORTIE FIN
     POUR X2
       I (2) '-' I (+1) DEN1 DE PERREC-REC
       I (+1) DEN2 DE PERREC-REC
       I (+5 -4) AN
       I (+3 -2) LUNA
       I (+3 -2) ZI
       ECRIRE
     FIN
     EXEC TOFIRST /* fixare pe urmatoarea recompensa */
     REFAIRE
   FIN
FIN ?
```

PROGRAMUL 1

```
.        DELETE DV:AD6,FN:'DEFIMT3 IMT'
.        ALLOC DV:AD6,FN:'DEFIMT3 IMT',AM:ANY,SZ:10
.        SYSRUN BIBLIO
BIBLIO STARTED

% OPNLIB A,DV:AD6,LN:DEFIMT3,VN:0,GN:1,FT:IMT,INIT

% ENDLIB

.      ,* COMPILE ASSIRIS

ASSIRIS 15.A 09/02/89 06.54

                    1 BCB       DSECT
                    2           RES,4     10
 00000028 F         3 ADCDRT1   RES,4     1
                    4           RES,4     34
 000000B4 F         5 TABXI     RES,4     10
 000000DC F         6 TABYI     RES,4     26

                    7 REFCONT   CSECT
                    8           USD,BCB   11
                    9 * PARAMETRI DE APEL LA REFACERE CONTOR <SIR DE BITI>
                   10 * X1::=ADRESA REALIZARII 1 DIN ENTITATE
```

* Programul 1 aparţine testului 24.

```
                   11 * Y1::=DEPLASARE CONTOR SIR DE BITI
                   12 * Y2::=DIMENSIUNE CONTOR
                   13 * Y3::=NUMAR DE BITI POZITIONATI LA I IN SIR
                   14 *
0000 602A0078      15            LD4I,0     PAVA
0004 B10A0028 F 16            LD4,1      *11.ADCDRT1
0008 01280000 A 17            LD1I,1     X'00'
000C 00160004 A 18            SB4,0      4
0010 605A0074   19            ST4,0      ADPAV
0014 300A00B8 F 20            LD4,0      11.TABXI+4
0018 310A00E0 F 21            LD4,1      11.TABYI+4
001C 00160004 A 22            SB4,0      4
0020 050A0000 A 23            LD4,5      0
0024 64390054   24            BAL,4      LIREC
0028 042A0020 A 25            LD4I,4     32
002C 330A00E4 F 26            LD4,3      11.TABYI+8
0030 0416000C A 27            SB4,4      3*4
0034 012A0000 A 28            LD4I,1     0
0038 04EB0400 A 29            SRL8,0     ,4
003C 300A00E8 F 30            LD4,0      11.TABYI+12
0040 04EB0000 A 31            SLL8,0     ,4
0044 605A007C   32            ST4,0      PAVA+4
0048 000A0014 A 33            LD4,0      5*4
004C 64390064   34            BAL,4      ECRIC
0050 80380020 A 35            BRU        *32
 00000054       36 LIREC      EQU        *
0054 610A0074   37            LD4,1      ADPAV
0058 8639001C A 38            DATA,4,4   X'8639001C'
005C 08030004 A 39            DATA,4,4   X'08030004'
0060 80380010 A 40            BRU        *16
 00000064       41 ECRIC      EQU        *
0064 610A0074   42            LD4,1      ADPAV
0068 8639001C A 43            DATA,4,4   X'8639001C'
006C 0C030007 A 44            DATA,4,4   X'0C030007'
0070 80380010 A 45            BRU        *16
0074            46 ADPAV      RES,4      1
 00000078       47 PAVA       EQU        *
0078 00000100 A 48            DATA,4,4   256
007C 00000000 A 49            DATA,4,4   0
0080 00000000 A 50            DATA,4,4   0
                51            END        REFCONT

*** 0000 ERREUR

ASSIRIS 15.A 09/02/89 06.54

*** DENSITE DE COMMENTAIRES 2

*** NOMBRE DE CONSTANTES D'ADRESSE DE TYPE RA:0

MODULE REFCONT TYPE P LONGUEUR 0088

PLUS HAUT NIVEAU D'ERREUR RENCONTRE N=0 (PAS D'ERREUR)

           LINK ER:4,DV:AD6,LN:DEFIMT3,VN:0,GN:1,FN:REFCONT
LINK STARTED

LINK 15.06.02 09/02/89 06H54M47S

SEGMENT REFCONT NO 1    IMPLANTATION 0
        MODULE REFCONT IMPLANTATION 0
                             LONGUEUR DU SEGMENT 88

LINK 15.06.02 09/02/89 06M54M47S
```

```
0 ERREUR EN EDITION DE LIENS
ADRESSE DE LANCEMENT          0
LONGUEUR PLUS GRANDE BRANCHE 88
LONGUEUR DU PROGRAMME EDITE  88

PLUS HAUT NIVEAU D'ERREUR RENCONTRE N=0 (PAS D'ERREUR)

NOUVEAU FICHIER CREE

NOM=REFCONT,NUMERO DE MISE A JOUR=1
.          FETCH DV:AD0,LN:BIBIMTSO,VN:1,FN:SOCBTCH
.          ASSIGN K,DV:AD1+AD2
.          OPTION CS'CF:59,DS:12000,SF:(,JNLB)'
.          RUN TIME:99,NL:50000,AD:0,0
SOCBTCH STARTED

NB CADRE FICH:59
%          SOC PN:ASE,PW:SOC,AN:1,BN:BDDPERS
%          RUN FN:M

** SOCRATE STARTED ** V1.5 LE:09/02/89 A 06.56.29.02.

1 :SUPEXP REFCONT ?
ERR.MACRO 301 NL 1 .. REFCONT
ERREUR EN MODE MACRO
%          RUN FN:M

** SOCRATE STARTED ** V1.5 LE:09/02/89 A 06.56.29.78.

1 :DEFIMT REFCONT ?
2 DV:AD6,FN:'DEFIMT3 IMT/REFCONT',VN:0,GN:1,UN:255
OK PROG IMT ENREGISTRE:REFCONT
%          SAVE
09/02/89 A 06.56.39.02. VOL:1 NUM.BLOC JRN:128 CKPT NUM:1
%          LOGOUT
```

PROGRAMUL 4

```
           COMPILE ASSIRIS

ASSIRIS 15.A 09/02/89 06.54

                    1 BCB       DSECT
                    2           RES,4     10
        00000028 F  3 ADCDRT1   RES,4     1
                    4           RES,4     34
        000000B4 F  5 TABXI     RES,4     10
        000000DC F  6 TABYI     RES,4     26

                    7 FROMLAST  CSECT
                    8           USD,BCB   11
                    9 * PARAMETRI DE APEL
                   10 * X1::=ADRESA REALIZARII CARE CONTINE INELUL
                   11 * Y1::=DEPLASARE INEL IN ENTITATE
                   12 * X2::=ADRESA ENTITATII REFERITE
                   13 *
                   14 *
0000 602A0078      15           LD4I,0    PAVA
0004 B10A0028 F    16           LD4,1     *11.ADCDRT1
0008 01280000 A    17           LD1I,1    X'00'
```

* **Programele 4 şi 5 aparţin testului 25.**

```
000C 00160004 A 18          SB4,0      4
0010 605A0074   19          ST4,0      ADPAV
0014 300A00B8 F 20          LD4,0      11.TABXI+4
0018 310A00E0 F 21          LD4,1      11.TABYI+4
001C 001E0004 A 22          AD4,0      4
0020 00270004 A 23          AD4I,0     4
0024 64390054   24          BAL,4      LIRE
0028 305A00BC F 25          ST4,0      11.TABXI+8
002C 80380020 A 26          BRU        *32
 00000030       27 LIRE     EQU        *
0030 610A0074   28          LD4,1      ADPAV
0034 8639001C A 29          DATA,4,4   X'8639001C'
0038 08030004 A 30          DATA,4,4   X'08030004'
003C 80380010 A 31          BRU        *16
0040            32 ADPAV    RES,4      1
 00000044       33 PAVA     EQU        *
0044 00000100 A 34          DATA,4,4   256
0048 00000000 A 35          DATA,4,4   0
004C 00000000 A 36          DATA,4,4   0
                37          END        FROMLAST

*** 0000 ERREUR

ASSIRIS 15.A 09/02/89 06.54

*** DENSITE DE COMMENTAIRES 2

*** NOMBRE DE CONSTANTES D'ADRESSE DE TYPE RA:0

MODULE FROMLAST TYPE P LONGUEUR 0050

PLUS HAUT NIVEAU D'ERREUR RENCONTRE N=0 (PAS D'ERREUR)

            LINK ER:4,DV:AD6,LN:DEFIMT3,VN:0,GN:1,FN:FROMLAST
LINK STARTED

LINK 15.06.02 09/02/89 06H54M47S
SEGMENT FROMLAST NO 1   IMPLANTATION 0
        MODULE FROMLAST IMPLANTATION 0
                        LONGUEUR DU SEGMENT 50

LINK 15.06.02 09/02/89 06M54M47S

0 ERREUR EN EDITION DE LIENS
ADRESSE DE LANCEMENT              0
LONGUEUR PLUS GRANDE BRANCHE 50
LONGUEUR DU PROGRAMME EDITE  50

PLUS HAUT NIVEAU D'ERREUR RENCONTRE N=0 (PAS D'ERREUR)

NOUVEAU FICHIER CREE

NOM=FROMLAST,NUMERO DE MISE A JOUR=1
```

PROGRAMUL 5

```
COMPILE ASSIRIS

ASSIRIS 15.A 09/02/89 06.54

                1 BCB       DSECT
                2           RES,4      10
 00000028 F 3 ADCDRT1       RES,4      1
```

```
                     4             RES,4       34
     000000B4 F  5 TABXI           RES,4       10
     000000DC F  6 TABYI           RES,4       26
                 7 TOFIRST         CSECT
                 8                 USD,BCB     11
                 9 * PARAMETRI DE APEL
                10 * X2:::=ADRESA REALIZARII DIN ENTITATE CARE CONTINE REFERIRE
                11 * Y2:::=DEPLASARE REFERIRE IN ENTITATE
0000 602A0078   12                 LD4I,0      PAVA
0004 B10A0028 F 13                 LD4,1       *11.ADCDRT1
0008 01280000 A 14                 LD11,1      X'00'
000C 00160004 A 15                 SB4,0       4
0010 605A0074   16                 ST4,0       ADPAV
0014 300A00B8 F 17                 LD4,0       11.TABXI+8
0018 310A00E0 F 18                 LD4,1       11.TABYI+8
001C 001E0004 A 19                 AD4,0       4
0020 00270008 A 20                 AD4I,0      8
0024 64390054   21                 BAL,4       LIRE
0028 330A00E4 F 22                 ST4,0       11.TABXI+8
002C 80380020 A 23                 BRU         *32
  00000030      24 LIRE            EQU         *
0030 610A0074   25                 LD4,1       ADPAV
0034 8639001C A 26                 DATA,4,4    X'8639001C'
0038 08030004 A 27                 DATA,4,4    X'08030004'
003C 80380010 A 28                 BRU         *16
0040            29 ADPAV           RES,4       1
  00000044      30 PAVA            EQU         *
0048 00000100 A 31                 DATA,4,4    256
004C 00000000 A 32                 DATA,4,4    0
0050 00000000 A 33                 DATA,4,4    0
                34                 END         TOFIRST

*** 0000 ERREUR

ASSIRIS 15.A 09/02/89 06.54

*** DENSITE DE COMMENTAIRES 2

*** NOMBRE DE CONSTANTES D'ADRESSE DE TYPE RA:0

MODULE TOFIRST TYPE P LONGUEUR 0088

PLUS HAUT NIVEAU D'ERREUR RENCONTRE N=0 (PAS D'ERREUR)

        LINK ER:4,DV:AD6,LN:DEFIMT3,VN:0,GN:1,FN:REFCONT
LINK STARTED

LINK 15.06.02 09/02/89 06H54M47S
SEGMENT TOFIRST NO 1    IMPLANTATION 0
        MODULE TOFIRST IMPLANTATION 0
                              LONGUEUR DU SEGMENT 50

LINK 15.06.02 09/02/89 06M54M47S

0 ERREUR EN EDITION DE LIENS
ADRESSE DE LANCEMENT        0
LONGUEUR PLUS GRANDE BRANCHE 50
LONGUEUR DU PROGRAMME EDITE  50

PLUS HAUT NIVEAU D'ERREUR RENCONTRE N=0 (PAS D'ERREUR)

NOUVEAU FICHIER CREE

NOM=TOFIRST,NUMERO DE MISE A JOUR=1
```

BIBLIOGRAFIE
pentru ambele volume

[ADIBA80] Adiba M., J. M. Andrade, P. Decitre, F. Fernandez, N. G. Toan — POLYPHEME: *An Experience in Distributed Database System Design and Implementation, Distributed Data Bases*, North-Holland, 1980.

[ADIBA79] Adiba M. ş.a. — POLYPHEME: Les premièrs conclusions, *Bulletin INRIA nr. 57/1979*, pag. 17—19.

[ACS82] Avram V., Cojocaru A., Sabău Gh. — Generarea de programe SOCRATE pornind de la structura bazei de date. Generaţia 1-a:
— versiunea 1: în volumul Lucrările celui de al III-lea simpozion „*Modelarea ciberneticǎ a proceselor de producţie*", A.S.E., 1982;
— versiunea 2: comunicare la *simpozionul ICI* '82.

[A1] Avram V. — *Consideraţii privind realizarea unui produs portabil SOCRATE, pentru definirea asistată de calculator a structurii conceptuale a bazei de date şi generarea automată a programelor aferente acesteia* (manuscris 1986).

[AF1] Avram V., Felea G. — *Creşterea eficienţei realizării sistemelor informatice prin automatizarea proiectării şi programării*. Centrul de Calcul al Ministerului Apărării Naţionale, 1982.

[ARGUS] ARGUS — *Manual de prezentare*,
— *Manual de utilizare*,
— *Manual de instalare şi operare*.
Institutul de Cercetare Ştiinţifică şi Inginerie Tehnologică pentru Tehnică de Calcul şi Informatică (ICSIT-TCI), Bucureşti, 1986.

[BARA] Bara I., Râureanu M. — *Experiment privind arhitectura unui sistem inteligent de gestiune a bazelor de date relaţionale*, ICSIT-TCI, 1986.

[BERKELEY] * * * *Proc. of 6th Berkeley Workshop on Distributed Databases Management and Computer Networks*, Berkeley, feb. 1982, S.U.A.

[BERLIN] * * * *Proc. of 2nd International Symposium on Distributed Data Bases*, Berlin, sept., 1982.

[BOSC] Bosc P., A. Chauffant, R. Trepos — FRERES: charactéristiques et utilisations, *Bulletin INRIA nr.* 57, 1979, pag. 14—16.

[CHUPIN] Chupin J. C. — Control Concepts of a Logical Network Machine for Data Banks, *Information Processing* 74, North-Holland, 1974, pag. 291—295.

[CLIO] * * * CLIO — *Systeme de gestion de bases de données*, Presentation Groupe SYSECA 1985.

[CODD] Codd E. F. — Relational Completeness of Data Base Sublanguages, *IBM RJ987*, 1972.

[COJOCARU] Cojocaru A. — Algoritmi de sinteză a relaţiilor în forma normală trei, *Raport ICI, TR-0282*, 1982.

[COJOCARU83] Cojocaru A. ş.a. — LAMBDA: An Architecture for a Distributed File System, 6th *International Seminar on DBMSs*, Matrafured, oct. 1983, Hungary.

[COJOCARU84] Cojocaru A., Lamor F. — *Comparative Survey on DDBMSs, WG-19 Workshop*, Dresda, feb., 1984, R.D.G.

[COJOCARU85] Cojocaru A. — Arhitectura sistemi upravlenia raspredelenimi bazami damih, şedinţa *Grupei de Lucru nr.* 19, *Praga*, nov. 1985, R.S.C.

[DATE87] Date C. J. — *An Introduction to Database Systems*, Addison-Wesley, vol. 1, 4th edition, 1987.

[DAVENPORT] Davenport R. A. — Design of Distributed Database System, The *Computer Journal C4*, 1, 1981,.

[DECITREa] Decitre P., André E. — *POLYPHEME project* — The DEM Distributed Execution Monitor, *Distributed Data Bases*, North-Holland, 1980, pag. 85—98.

[DECITREb] Decitre P., Andrade J. — *Technical outline of the Polypheme demonstration prototype, Distributed Data Bases*, North-Holland, 1980, pag. 357—360.

[DELOBEL] * * * *Distributed Data Bases*, North-Holland 1980 (ed. C. Delobel, W. Litwin).

[DRAGHICI] Drăghici M. — *Îndrumar metodologic pentru proiectarea schemei conceptuale a bazei de date* — ICSIT-TCI, 1986.

[PCDPS] Pescaru V., Catona I., Duţă I., Popescu C., Satran I. — *Fişiere, baze, bănci de date*, Editura tehnică, 1976.
[DU82] Dumitrache V., Unghianu M. — *Băncile de date*. Editura ştiinţifică şi enciclopedică, 1982.
[GAILLARD] Gaillard M. — *Contribution à la conception, la réalisation et l'utilisation du systeme de bases de données* SOMINE.
[GARDARIN82] Gardarin G., Melkanoff M. — Concurrency Control Principles in Distributed and Centralized Databases, *INRIA Report no.* 113, 1982.
[GARDARIN84] Gardarin G. — *Bases de donnees, Les systemes et leurs langages*, Dunod, 1984.
[IACS83] Ionescu C., Avram V., Cojocaru A., Sabău Gh. — Generarea de programe SOCRATE pornind de la structura bazei de date. Generaţia 2-a, în volumul *Lucrările celui de al IV-lea simpozion „Modelarea cibernetică a proceselor de producţie"*, ASE, 1983.
[INFOTECH] * * * Distributed Databases, *Infotech State of the Art*, 1979, 2 vol.
ISCA85[] Ionescu C., Sabău Gh., Cojocaru A., Avram V. — *Baze de date relaţionale*. Lito A.S.E., 1985.
[JARDINE] Jardine D. A. — *The ANSI/SPARK DBMS Model*, North -Holland, 1978.
[KING] King J. — *Evaluating Data Base Management Systems* Von Nostrand Reinhold, 1981.
[LDA] LEDA — *Manual de utilizare şi operare*. ICSIT-TCI, Bucureşti, 1986.
[LINDSAY] Lindsay B., Selinger P. G. — Site Autonomy Issues in R* : A Distributed Database Management System, *IBM Research Report RJ* 2527, 1980.
[MATY] Maty P. — *Contribution à la conception, la réalisation et l'utilisation du systéme de bases de données* SOMINE.
[MIDAS] * * * *SOCRATE-mini*, V3, CTCE Constanţa, 1987.
[MOHAN] Mohan C., Lindsay B., Obermarck R. — Transanction Management in the R* Distributed Database Management System, *ACM TODS*, 11, 4, 1986, pag. 378—396.
[P1] PACSIN *Manual de prezentare*, ICSIT-ITCI, Bucureşti, 1984.
[SABAU] Sabău Gh., Avram V., Lungu I. — *Sisteme de gestiune a bazelor de date — sistemul SOCRATE, prezentare şi aplicaţii*, Lito A.S.E., 1982.
[SAYETTAT] Sayettatt C. — *Contribution à la conception, la realisation et l'utilisation du sysètme de bases de données* SOMINE.
[SI] SOGER, *Manual de prezentare*, Centrul de Calcul al C.S.P., 1982.
[S2] J. R. Abrial, J. Bas, G. Beaume, C. Herneron, R. Morin, G. Vigliano, *Project SOCRATE*, Universite de Grenoble, 1970.
[S3] SOMINE, Contribution a la conception, la realisation et l'utilisation du systeme de bases de donnees :
— C. Sayettat : *Interface entre SOMINE et ses utilisateur et aide à la conception asistée par ordinateur* ;
— P. Marty : *Structuration des informations dans SOMINE et recherche des structures optimales* ;
— M. Gaillard : *Gestion des memories de SOMINE et enseignement assisté par ordinateur*. Institut National Polytechnique de Grenoble, 1976.
[STEEL] Steel T. — Data Base Standardisation : *A Status Report, Data Base Description* (ed. B. Dougne şi G. Nijssen), North-Holland, 1975.
[STONEBRAKER] Stonebraker M. — *Concurrency Control and Consistency of Multiple Copies of Data in Distributed* INGRES, IEEE TSE SE-5, 3, 1979.
[ULLMAN] Ullman J. D. — *Principlés of Database Systems*, Computer Science Press, 1980.
[V1.6.76] * * * Ameliorations apportées par la V1.6, Tome 1, *ECA Automation*, 1976.
[V1.6.77] * * * Ameliorations apportées par la V1.6, Tome 2, *ECA Automation*, 1977.
[V1.6.78] * * * Ameliorations apportées par la V1.6, Tome 3, *ECA Automation*, 1978.
[V1.7.76] * * * SOCRATE V1.7. *Manual de presentation*, *ECA Automation*, 1976.
[V1.8.77] * * * Presentation des ameliorations SOCRATE pour une V1.8, *ECA Automation*, 1977.
[ZURLUH] Zurluh G. — *Le projet PLEXUS* — Etude et réalisation d'un système transactionnel de gestion de bases de données réparties, teză de doctorat 1981.

• SOCRATE este sistemul de gestiune al bazelor de date pe calculatoarele româneşti Felix C, provenit de la familia franceză IRIS-50. Acest SGBD este cunoscut şi disponibil de mult timp în toate unităţile de calcul electronic din ţară. SOCRATE-MINI este un SGBD de acelaşi tip, realizat în ultimii ani de specialişti ai Centrului Teritorial de Calcul Constanţa pentru minicalculatoarele CORAL/INDEPENDENT şi cele compatibile cu familia americană PDP 11.

• Cartea îmbină preocupările a două grupuri de specialişti : primul compus din cadre didactice, analişti şi programatori de sistem şi aplicaţii de la Academia de Studii Economice, Institutul de Tehnică de Calcul şi Informatică, Comitetul de Stat al Planificării şi al doilea — din elaboratori ai produsului-program SOCRATE-MINI.

EDITURA TEHNICĂ

• Au rezultat două volume care îşi merită subtitlul „Totul despre...", deoarece, pe de o parte, SOCRATE, acest SGBD — de tip reţea, mai complex decît unul relaţional — este tratat complet, pornind de la limbajele de descriere şi de manipulare a datelor, copios exemplificate, pînă la optimizări şi dezvoltări SOCRATE şi alte SGBD-uri, iar pe de altă parte, SOCRATE-MINI este reprezentat prin manualele de lansare-utilizare-operare pînă la ultima versiune (V 4.0), cu multiple aplicaţii. Şi nu numai atît ; fiecare volum se încheie cu cîte un studiu de caz, în extenso : o aplicaţie de personal, respectiv una de transport naval.

• Datorită acestei structurări, spectrul celor interesaţi este deosebit de larg : informaticieni, cadre didactice, studenţi, utilizatori actuali şi potenţiali de baze de date, din toate domeniile economiei naţionale.

ISBN 973-31-0020-X
ISBN 973-31-0021-8

VOL. 1 + 2 Lei 62